Robot Instruction by Human Demonstration

Sing Bing Kang

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the field of Robotics

Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

December 1994

This research was sponsored in part by the Advanced Research Projects Agency under the Department of the Army, Army Research Office under grant number DAAH04-94-G-0006, in part by the Advanced Research Projects Agency under the U.S. Air Force, the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), Wright-Patterson AFB under Contract F33615-90-C-1465, ARPA Order No. 7597, and in part by National Science Foundation under Contract CDA-9121797. Views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of the United States Government.

Abstract

Conventional methods for programming a robot either are inflexible or demand significant expertise. While the notion of automatic programming by high-level goal specification addresses these issues, the overwhelming complexity of planning manipulator grasps and paths remains a formidable obstacle to practical implementation. This thesis describes the approach of programming a robot by human demonstration. Our system observes a human performing the task, recognizes the human grasp, and maps it onto the manipulator. Using human actions to guide robot execution greatly reduces the planning complexity.

In analyzing the task sequence, the system first divides the observed sensory data into meaningful temporal segments, namely the *pregrasp*, *grasp*, and *manipulation* phases. This is achieved by analyzing the human hand motion profiles. The features used are the fingertip polygon area (the fingertip polygon being the polygon whose vertices are the fingertips), hand speed, and the *volume sweep rate*, which is the product of the first two. Segmentation is achieved by taking into consideration the hand motion profiles during the pregrasp (or reaching) phase having a characteristic bell shape.

Subsequent to task segmentation, a grasp taxonomy is used to recognize the human hand grasp. The grasp taxonomy is based on the *contact web*, which is a 3-D graphical structure of contact points between the hand and the grasped object. By considering the higher level concept of the *virtual finger*, which is the collection of physical fingers acting in a similar manner, we can recognize the type of human grasp used in the task.

The recognized grasp is used to guide the grasp planning of the manipulator. Grasp planning is done at two levels: the functional and physical levels. Initially, at the functional level, grasp mapping is achieved at the virtual finger level. Subsequently, at the physical level, the geometric properties of the object and manipulator are considered in fine-tuning the manipulator grasp. The trajectory of the manipulator approximately follows that of the human hand during the execution of the task. Once all these are accomplished, control signals are then produced for the robot system to replicate the task.

In summary, this thesis describes a novel way of programming robots – by direct human demonstration of the task. This thesis shows that by segmenting the stream of observed data and producing abstract representations of the task, we can enable the robot to replicate human grasping tasks.

Acknowledgments

I would like to express my gratitude to Katsushi Ikeuchi for being my advisor and mentor. He has taught me what it takes to be a productive and methodical researcher. I am also grateful to Michael Erdmann, Takeo Kanade, Matthew Mason, and Kenneth Salisbury (of Artificial Intelligence Lab., MIT) for being in my thesis committee and for providing very useful feedback on my work.

Bob Harper and Steve Shafer were kind enough to offer advice and encouragement during my job search. Jon Webb wrote the iWarp code for the multibaseline stereo system used in my thesis work. I also appreciate Richard Szeliski's (of Cambridge Research Lab., DEC) guidance while I was a summer intern at DEC in 1992.

I am grateful for the friendship and company of the Cantonese-speaking "gang of four," namely Harry Shum, Jeffrey Yang, Isaac Levin, and David Chan. There are many other people whom I would like to acknowledge for their help in one way or the other, from lending a sympathetic ear to helping me with my thesis work: Ashraf Ali Kassim, Lin Chase, Herman Herman, Kazunori Higuchi, Masato Kawade, Jin-Oh Kim, Kyoung (Harry) Kim, John Krumm (especially for his helpful discussions on the spectrogram), Dan Morrow, Brad Nelson, Carol Novak, George Paul, Yoichi Sato, David Simon, Fred "Prophet of Doom" Solomon, Richard Voyles (especially for his help with the PUMA arm and Utah/MIT hand), Mark Wheeler (especially for his 3DTM code for object localization), and Reg Willson. CMU life would have been less interesting if not for my former officemates Chris Colby, David Eckhardt, and John Greiner.

Finally, I would like to thank my family for their love, support, and encouragement, especially Ayeh (whom I shall always miss dearly and to whom I dedicate this thesis), Sing Chye, Julie, Ratna, and Soon Kee.

Table of Contents

Chapter 5

Chapter 6

Chapter 7

List of Figures

List of Tables

Chapter 1

Introduction

Automation is playing an increasingly important role in industry. It keeps companies competitive by increasing product quality and output. To meet the possible demands of rapidly changing consumer needs, robots used in automating tasks must be designed for flexibility and fast reprogrammability. The ease of programming a robot and the amount of time required to do so are thus significant issues in robot system design for integration into the manufacturing process.

Robot programming is the act of specifying actions or goals for the robot to perform or achieve in order to carry out a useful task; as such, it is an essential and necessary component of task automation. It can be subdivided into several different categories primarily according to the type of data input (namely low-level sensor signals, mid-level robot instructions, or high-level goal specification). A good description of such a set of categories is given by Lozano-Pérez [90]. We give a brief description of our categorization of robot programming methods here.

1.1 Robot Programming Approaches

The robot programming methods are *teleoperation, teaching, robot-level textual programming*, and *automatic programming*. The teaching and robot-level textual programming methods are by far the most pervasive in both the industrial and academic environments. Note that these robot programming categories are not mutually exclusive and that a given programming approach may be a combination of these categories.

1.1.1 Teleoperation

Teleoperation, in the simplest sense, is not really a programming approach; it basically refers to the direct control of a remote manipulator (the slave) by manipulating a master device. As in teaching, which is described shortly, the input data are low-level sensor signals; while input data is stored for repeated execution in teaching, they are immediately

(ignoring link delays) used to move a remote manipulator. Teleoperation can thus be viewed as a teaching method with concurrent execution (albeit with link delays). Control signals are normally not stored and replayed because teleoperation is usually performed in unstructured environments, where the environment may be unknown or consequences to actions may be unpredictable or difficult to model; these kinds of environment require continual monitoring in order to react appropriately to possible changes. This is in contrast to teaching, where the environment is known and structured, with consequences to actions predictable — thus playing back stored data repetitively does not present problems.

Since teleoperation is specialized and works with low-level sensor signals, the problem of non-portability across different robot systems exists. However, the element of risk of injury to the operator by the robot is very much diminished compared to teaching methods (especially in the case of teaching by direct manual guidance, or *lead-through teaching*). In addition, since the slave manipulator is very likely to be kinematically dissimilar from the human hand, controlling its degrees of freedom is not very intuitive and thus require some amount of training.

Teleoperation is used in applications that requires remote handling of hazardous material or operations in unfriendly environments. Research in teleoperation is actively pursued in applications at nuclear sites and in space and underwater; in addition, it is also being used to help the disabled. Relatively recent developments have resulted in different levels of sophistication in teleoperation, e.g., force magnification, force reflection, some degree of autonomy to cater for delays in remote links, and preprogrammed motions. These developments increasingly blur the distinction between teleoperation and other robot programming approaches.

1.1.2 Teaching Methods

In teaching methods (e.g., [5], [35], [80]), the robot or manipulator learns its trajectory either through a teaching pendant or actual guidance through a sequence of operations. Here, the input data are sensor signals; they are recorded and stored subsequent to teaching. The manipulator executes the task by playing back this stored sequence of data. Using the teaching pendant to program a robot has been referred to as *walk-through teaching* while physically guiding the manipulator through the desired path is called *lead-through teaching* (the latter is also known as “teach-by-guiding,” or less appropriately “teach-by-showing”). The teaching method is the easy to use since implicit knowledge of the task is not necessary. On the other hand, because teaching methods involve some degree of repetition by the human operator owing to errors, it can be tedious and tiring. In addition, the robot may pose some risk of injury to the operator if he/she is teaching the robot within its workspace (especially for lead-through teaching). Other disadvantages of this method are that its stored data are not easily modified for a different task and they are not easily transferable to a different

system. In addition, interpretation of the data may also be difficult due to the presence of sensor noise.

1.1.3 Robot-level Textual Programming

Robot-level textual programming (e.g., [38], [43]) refers to the approach of hand-coding programs to enable the robot to execute motions in the robot joint and task space. While this approach of robot programming is flexible, it requires expertise and often a long development time. Several of the robot programming languages include AL, AML, RAIL, RPL, and VAL. A summary of these programming languages can be obtained from Gruver, *et al*. [43]. There has been some work done on reducing the amount of tedium associated with hand-coding and debugging by incorporating more user-friendly facilities to interact with the robot system. For example, Summers and Grossman [142] combine teaching and interactive dialogue with the system to facilitate the incorporation of sensor strategies in their XPROBE system. XPROBE would then generate AML code for the performance of the desired task.

1.1.4 Automatic Programming

The problems associated with teaching and textual programming can be alleviated by automatic programming (e.g., [55], [78], [91], [149]), also known as *task-level programming* [90]. In automatic programming, conceptually, the only inputs to the robot system required to generate the control command sequences are the description of the objects involved in the task, and the high-level task specifications.

Realization of a practical system with complete automatic programming is difficult since important issues remain relatively unresolved in a satisfactory manner. Such issues include: How does one generate a sequence of operations? How can tasks be described unambiguously? If a task involves grasping, how can a stable grasp be effected — should it be optimal from the "human" point of view or a purely analytic point of view? While research by Lozano-Pérez, *et al*., on the HANDEY simulation system (with a parallel-jaw gripper mounted on a PUMA arm) [92] and Laugier and Pertin-Troccaz on the SHARP system [82] are positive steps in this direction, there is still work to be done on automatic planning with dextrous hands.

1.2 What is This Thesis About?

This thesis looks at a method of programming a robot, specifically programming by human demonstration. The key idea of this approach is to dispense with the bulk of tedious hand-coding and allow the robot to be programmed by having it watch a human operator perform the task. The inputs are low-level sensor signals which are interpreted to produce higher-level descriptions of the task; this then enables the system to understand the task to the extent of being able to accomplish it. As a result, this approach is a mix of teaching and

automatic programming — we collectively label this hybrid programming approach as *programming by demonstration*.

This thesis is motivated by our philosophy that favors minimal programming effort while maintaining the desired robot performance. A robot programming system should be judged not only by its performance, but also by how easy it is to program the system. A good robot programming system is one in which the amount of programming skill required of the operator is kept to the minimum. We want to simplify the programming process at the operator level as much as possible, even if it means designing a more complex input interpreter[1]. The simplest manner one can think of in which an operator can program a robot is to demonstrate the task directly in front of the system and expect the system to replicate the task with little other human intervention.

This thesis shows that a programming-by-demonstrating system which is capable of:

1. *Recording a perceptual data stream during the human performance of the task,*
2. *Analyzing the perceptual data stream,*
3. *Producing task descriptions, including the description of the human grasp, and*
4. *Executing the task*

is realizable.

1.3 Organization of Thesis

This thesis is organized primarily according to the constituent modules that compose our programming-by-demonstration system.

In Chapter 2, we describe the programming-by-demonstration approach and reiterate our justification in choosing this approach in robot programming. An overview of our system is also given in this chapter.

A very important component in the programming-by-demonstration system is the observation or data acquisition module, which is described in Chapter 3. This module is responsible for acquiring the perceptual stream of data associated with the human execution of the task for analysis and task replication. The importance of data quality extracted by the data acquisition module cannot be overemphasized, as it dictates the amount, complexity, and type of processing required to yield the desired task description.

---

1. In other words, life should be made much easier for the operator or user but much harder for the system designer.

Once the perceptual stream of data is acquired, it is analyzed for the purpose of task replication. However, in order to produce task descriptions, it is easier to divide this perceptual stream into meaningful time segments for individual analysis. Chapter 4 describes how this is done and what these meaningful segments are. A very important segment of the perceptual data stream is the segment that corresponds to the time when the object is grasped.

Grasp synthesis is a hard problem, especially for dextrous manipulators. The approach that we use considers human actions taken during task execution as a guide to robot planning, including that of the grasp. As such, the process of recognizing the human grasp employed in the task is pivotal to our robot task planning. Chapter 5 shows how a human grasp can be identified. The grasp recognition process uses the *contact web* representation of the grasp (3-D arrangement of contact points between the hand and grasped object) and a grasp taxonomy based on the contact web.

Chapters 4 and 5 describe the major processes that constitute the functions of the task recognition module. Chapter 6 illustrates, with several examples, how tasks are analyzed from temporal task segmentation to grasp recognition. It also details how object motion can be extracted and repetitive manipulative motions such as turning a screw can be detected.

Once the task descriptions have been extracted, they are used to plan the manipulator grasp and motions. This is the job of the task translator. The task translator, by using this extracted information as well as robot kinematic and geometric information, produces robot commands to the robot system. Both the task translator and the robot system used as a testbed for our ideas are described in Chapter 7.

Chapter 8 illustrates the entire programming-by-demonstration system with an example task, from recording the human execution to replicating the task using the robot system. The techniques in analyzing the perceptual data stream and planning the robot execution of the task are detailed for this example task.

A summary of the system with concluding remarks are given in Chapter 9. Possible directions that can be taken to increase the level of sophistication of the programming-by-demonstration system are listed in this chapter as well.

Chapter 2

Programming by Demonstration

In the previous chapter, we have briefly described the various approaches to programming a robot and outlined some of the problems associated with these approaches. We could at least mitigate such problems by using a different approach to task programming. Our philosophy in designing a robot programming system is to simplify the programming process at the operator level as much as possible, even if it means designing a more complex input interpreter. The simplest manner an operator can program a robot is to demonstrate the task in front of the system and expect the system to replicate the task with little other human intervention. This is the method that we adopt in task programming; it is also known as the *Assembly Plan from Observation* (APO) approach which has been proposed by Ikeuchi and Suehiro [58][141]. In APO, task programming is performed by demonstrating the task to the system rather than by the more traditional method of hand-coding. The key idea is to enable a system to observe a human perform a task, understand it, and then execute the task with minimal human intervention. The realization of such a system requires the understanding of hand grasping motions. This method of task programming would obviate the need for a programmer to explicitly describe the required task, since the system is able to understand the task based on observation of the task performance by a human.

An approach similar to APO was taken by Kuniyoshi, *et al*. [78], who developed a system which is capable of emulating the performance of a human operator using a real-time stereo vision system. However, their system is restricted to pick-and-place operations and fingertip grasps. In order to replicate observed tasks, our system first infers intended subgoals from observation of actual task performance. Yared and Sheridan's work [158] is relevant in this respect; their system infers human intention by computer linguistic analysis of sequences of textual robot commands.

The APO approach of programming a robot belongs to a class of what we term as a *programming-by-demonstration* approach. This class of robot programming approach involves a user-friendly interface that allows the user to easily specify the task in a simple and intui-

tive manner without resorting primarily to hand-coding programs. The teaching and teleoperation methods described in the previous chapter are subsets of this approach.

2.1 Programming by Demonstration

Encoding knowledge about the task can be time-consuming; teaching a robot by demonstrating the task by the human operator helps simplify the task knowledge transfer process. This motivation is evident in work such as those involving learning human motor skills through observation [50], and recognizing pick-and-place operations through observation [78]. The programming-by-demonstration approach to robot programming can be roughly divided into five categories (which are not necessarily mutually exclusive):

2.1.1 Direct Motion of Robot Hand

In this category, the robot is taught by directly moving it through the desired trajectory. Obviously, the teleoperation and teaching methods described in the previous chapter belong to this category. In addition to merely using raw sensor data to playback the task, more recent work include robot motion interpretation, such as identification of robot states during assembly. In their effort to analyze and duplicate human manipulation of objects, Takahashi, Ogata, and Muto [145] use the approach of directly moving the object attached to the manipulator and interpreting the positional data. The data is translated into directional information and assembly states, the latter of which is based on relative position of the manipulated object to the environmental objects. By using joint and force sensor information from the robot, the system developed by Hirai and Sato [48] is capable of recognizing certain robot actions, such as approach, move-to-grip, and grip actions. The recognition of teleoperated robot motions is based on specified rules with pre- and post-conditions that are functions of sensor data and possibly other robot motions. For example, an approach motion is deemed to have occurred if the pre-conditions of the existence of the object and the hand being empty, as well as the post-condition that the hand is near the object, have been satisfied.

Pook and Ballard [124] use significant changes in manipulator finger tension as a means to detect completion of a subtask. In addition, they use the pattern classification technique of learning vector quantization to classify teleoperated manipulation. In their example task of placing a spatula under an egg on a pan, the classifications are grasp, carry, press, and slide. They also work on what can be classified as symbolic-level teleoperation or “deictic teleassistance,” [123] where a teleoperator uses hand signs to guide a manipulator through a given task. A hand sign may indicate reach, grasp, or turn actions, for example.

2.1.2 Using Physical Teaching Device

One of an earlier work on teaching by demonstration involves what is called a “visual programming device” (VPD) [134]. A VPD is a wand with an array of lights at one end, and is used in conjunction with a camera. The VPD is used to indicate the robot hand position and orientation as well as its trajectory. While it is easy to use, there are problems with repeatability.

Grossman and Taylor [41] propose using a mechanical deformable pointer (“bendy pointer”) to indicate position. By using a series of locations, the poses of object frames can be extracted easily and the code produced automatically. In an extension, the XPROBE system [142] uses the dialogue approach to interact with the operator while guiding the robot through a sequence of desired motions. It also allows the operator to teach sensing strategies associated with object detection, such as the time of activation of the infra-red beam sensor to detect an object prior to grasping.

Delson and West [35] measure position and force during the human execution of the task using a teaching device. This device comprises a gripper fitted with pressure sensors and a 3-D position sensor. In the work that they reported, they use the human variation in the repeated demonstrated trajectories to identify the shortest trajectory within the resulting obstacle-free region. Their analysis is restricted to planar translational motion, and does not use any geometric models of the task. They later describe an alternate heuristic-based local method to produce a synthesized 3-D robot trajectory [34]. In an earlier related work, Harima and West [45] use a dataglove with ultrasonic transducers (PowerGlove) to record the desired motions, which are subsequently piecewise fitted with line segments.

2.1.3 Using Virtual Reality Environment

Takahashi and Ogata [144] use the virtual reality (VR) environment as a robot teaching interface. The operator’s movements in the virtual reality space via the VPL dataglove are interpreted as robot task-level operations by using a finite automaton model. For example, an object is deemed to be grasped in the VR environment if the fingers are flexed around the VR object.

While Takahashi and Ogata investigate the transfer of gross human motion in the VR environment to robot motion, Hashimoto and Buss [46] concentrate on static grasp analysis of the human hand while interacting with objects in VR environment. They model the manipulative skill using a time-based sequence of the grip transformation matrix proposed by Salisbury [101]. The interface that they use (which they call the “sensor glove”) measures finger joint positions and provides force feedback. The sensor glove, which is affixed to a vertical metallic circular frame, limits the motion of the human hand, however. Interaction between the human hand (through the sensor glove) and the VR object results in force feedback at the

sensor glove. The grasp analysis performed involves both joint positions measured and torques synthesized as a result of manipulating the VR object.

2.1.4 Vision-based Programming

Inaba and Inoue [59] propose "vision-based robot programming," which refers to the task specification and programming via a computer vision interface. Task specification involves initializing object models using a 3-D pointer (laser spot scanner); task programming is subsequently done by "moving" the established object model to the desired position in the vision interface. Feature trackers in the form of rectangular windows are also specified by the user to allow the system to monitor the execution of the task. In line with this idea, Kuniyoshi, *et al.* [78] make use of a real-time stereo tracking system to determine pick-and-place motions.

Similar to Inaba and Inoue's work [59], Hamada, *et al*. [44] propose a system in which the operator selects objects from a menu displayed on a computer screen and "attaches" it to the image of a real scene, where it is superimposed onto the scene as a wireframe object. The task is achieved by specifying robot motion in the screen image using a mouse. Grasping points on object are known *a priori*. Commands such as "carry(cap, path, body)" are specified to interactively carry out operations. This is first simulated in a "task mental image" comprising *a priori* action knowledge and a graphical display. Subsequently, the operations are carried out by the manipulator with the aid of a computer vision system that matches the "mental image" models with the real objects.

Matsui and Tsukamoto [103] describe a system to facilitate task programming using a "multi-media display," which comprises a stereo pair of camera and graphics images (viewed with stereoscopic glasses for 3-D effect) and multiple windows for text display and menus. Objects are modeled by their boundary representations. Prior to task programming, the object models are superimposed onto real objects by first coinciding their reference frames before adjusting object orientation and translation. The task is then programmed by concatenating a series of operations from the operation menu. Subsequently, the task is simulated to check for possible manipulator-object interference before the task is actually carried out. Superimposing the expected scene (manipulator and objects) to the actual scene is done during the execution of the task for ease of operator verification.

Shepherd [135] describes a visual robot programming environment which allows interaction with both constructed scenes (using a mouse to position known object models) and actual scenes captured using a camera. The demonstrated task is represented as a collection of sensing and action rules. Each rule is taught by demonstrating first its preconditions and then its actions. He uses the term "visual programming" in a sense of program specification in a two or more dimensional fashion [111]. (Conventional textual languages are one-dimensional.) This approach is used to also teach the robot the sensing strategy via an interactive

dialogue in which the operator shows the operation using a mouse and the system responds with more detailed questions. In addition, the system uses more realistic scenes by a "cut-and-paste" method of objects shown during teaching at various poses.

2.1.5 Natural Programming

This type of robot programming entails fitting the programming interface to the operator, thus reducing the programming effort to just interpreting human motions. Harima and West [45] use the term "natural robot programming" to refer to the use of natural human motion to program a robot. They use a PowerGlove dataglove to indicate discrete 3-D trajectory points, which are subsequently used to generate the robot trajectory. In the same vein, Delson and West [35] use human hand trajectory (assumed planar) to plan manipulator trajectory.

Kuniyoshi, *et al.*'s robot programming system [78] recognize pick-and-place operations using a real-time stereo system. Two other restrictions of their system are that the object has to be picked by the fingertips and that the hand has to be oriented such that it is seen from the side by the camera system.

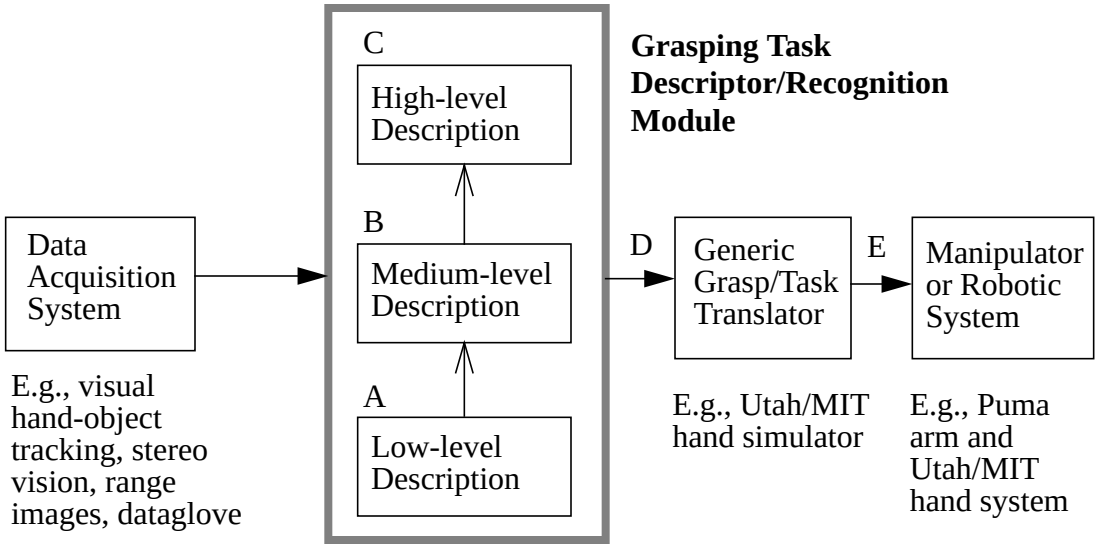
Our approach falls neatly into this category as well, since it involves the interpretation of natural human hand motion to program a robot. In contrast to other work, our system has the ability to recognize different types of human grasps in order to plan the manipulator grasp accordingly. Our approach directly follows Assembly Plan from Observation (APO) [58][141].

2.2 Assembly Plan from Observation

As mentioned in earlier sections, the key idea of *Assembly Plan from Observation* (APO) is to enable a system to observe a human perform a task, understand it, and perform the task with minimum human intervention. In this approach, the human provides the intelligence in choosing the hand (end-effector) trajectory, the grasping strategy, and the hand-object interaction by directly performing them. This approach helps to alleviate the problems of path planning, grasp synthesis, and task specification. Our system, which incorporates the APO paradigm, is shown in Fig. 1.

The data acquisition system extracts perceptual data from the environment; it provides low-level information on the hand location and configuration, objects on the scene, and with some analysis, contact information between the hand and the object of interest. Note that vision need not necessarily be the sole sensing modality through which low-level data is extracted. The grasping task descriptor/recognition module constitute a major portion of our work. It translates low-level data into higher levels of abstraction to describe both the motion and actions taken in the sequence of operations performed in the task. The vertical

arrows in this module as shown in Fig. 1 indicate the consolidation and interpretation of lower-level information to yield successively higher-level information. The low-level description refers to the joint angles and positions, the medium-level the virtual fingers and opposition space (defined in Chapter 5), and the high-level the type of the grasp itself.



A, B, C: Hierarchical description modules of grasping task
D: Output description of grasping task
E: Control commands

Fig. 1 **System incorporating principles of Assembly Plan from Observation**

In our system, the output of the grasping task descriptor module is subsequently provided to the task translator which in turn creates commands for the robotic system to execute in order to carry out the observed task. The representations given in submodules A, B, and C are independent of the robot manipulator (which is part of the robot system), while this is not true of the translator. Fig. 2 illustrates succinctly how our programming-by-demonstration system works. It analyzes the stream of perceptual data associated with the task by recognizing the human grasp before planning and executing the manipulator grasp.

Our work in this area is expected to result in a greater understanding of grasping motions, to the extent that recognition by a robotic system would be possible. The areas in which this body of knowledge is potentially useful include planning and automation, and teleoperation. Specifically, in our work, since the grasp is described using increasingly abstract and manipulator-independent representations, the resultant system would be conceptually applicable to any given manipulator to be used in the robotic system. One potential problem is that the best grasp is dependent on both the relative shapes and sizes of the object and hand. This is easy to see by comparing the grasps that are used to secure a hold on a medium-sized object and a small object. We are using the assumption that the relative sizes of the hand and

manipulator are comparable, so that an object that can be held comfortably within the compass of the human hand can also be held in a similar manner using the manipulator. If the manipulator is of a disproportionate size relative to the hand, it would not be difficult to scale the size of the object appropriately. However, the dynamics and control issues would not be as simple.

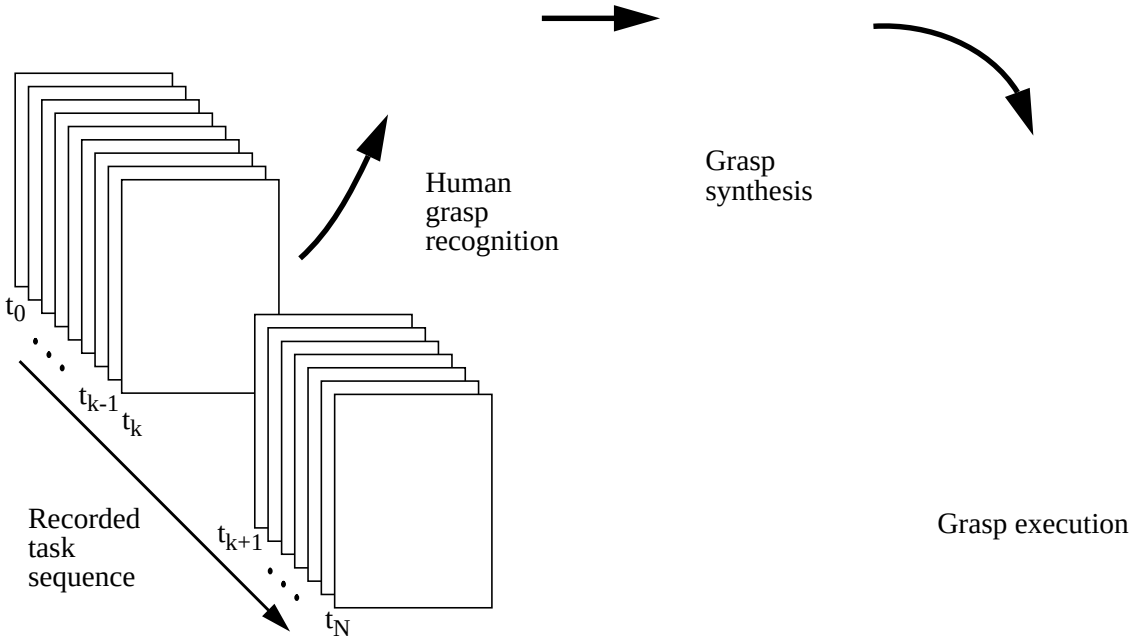


Fig. 2 **Operations in our programming-by-demonstration approach**

It is probably not true in general that the way the human performs the task is the best way for the robot as well. What we seek in our work is the convenience in programming the task, with the primary aim of relieving the human of the more mundane job of programming using a robot programming language (which also requires some degree of expertise). The proposed approach of APO may be done at the expense of optimal robot performance, but the savings in programming time and lower requirements of programming expertise would offset this possible shortcoming. In addition, the resulting system would be more portable from one robotic system to another, since the high-level abstractions of the grasp is manipulator-independent.

2.3 Using Explicit Knowledge of the Task

Our approach is not to force the operator to explicitly define the task. The system relies only on data extracted during the execution of the task itself. Explicit information (such as the

type of object manipulation to be performed) would definitely constraint certain aspects of the task – e.g., the type of grasps that one may use – but the rules governing such constraints are usually heuristic (e.g., [29] and [55]). We choose to rely directly on demonstrated human cues as perceptual inputs (as opposed to textual) to the system[1]. The operator transfers the task specification implicitly. The idea is not to expect him/her to provide explicit task specification[2] (which may not be easy to provide all the time - for example, how does one consistently specify the desired mobility and dexterity?).

While direct knowledge of the object geometry would be useful in identifying potential grasps and fingertip location, we did not use this approach, since the direct human demonstration would provide the necessary information of object face contact. However, adding object geometrical knowledge would definitely help (for example, to corroborate the identity of the grasp).

1. This is not too different from the idea of using constraints; in our system, we use perceived and recognized human cues to constrain (namely identify) the type of grasp and manipulative motions.
2. While it is not unreasonable to make use of explicit task specification when available, we are primarily interested in the use of perceptual data from human demonstration in programming a robot.

Chapter 3

Observation System

The observation or data acquisition system is responsible for collecting the stream of perceptual data associated with the operator performance of the task which is to be replicated by the robot. At the very least, the data acquisition system should provide hand location and posture as well as object location at every sampled instance during task execution. We have used two versions of this system. In the first version, the hand is tracked using a dataglove; object location is estimated from range images taken just prior to and right after task execution. The location of the object subsequent to the task performance is inferred both from human hand motion and an object localization program. In the second version of the data acquisition system, we replace the slow light-stripe rangefinder with a four-camera and iWarp system, which is capable of appreciably faster rates of image capture.

3.1 Setup Version 1

The observation data of the task performance is captured using a light-stripe rangefinder, a CCD camera (which yields the range and intensity images respectively), and a *Cyber*Glove with a Polhemus device (which yields the hand joint angles and pose respectively). The observation system is shown in Fig. 3. We track the configuration (joint angles) and pose (position and orientation) of the hand using the *Cyber*Glove [33] and Polhemus [1] devices respectively. The *Cyber*Glove is an instrumented, lightweight, flexible glove produced by Virtual Technologies. It has 18 sensors (3 flexion sensors for the thumb and 2 for the other fingers, 4 abduction sensors, 1 pinky rotation sensor, and the wrist pitch and yaw sensors). The distal interphalangeal joint angles are not measured but estimated instead based on the theoretically derived and empirically tested relationship between the distal and proximal interphalangeal joint angles [21]. The Polhemus 3Space Isotrak sensing device is attached to the back of the hand[1], and provides the position as well as the orientation of the hand rela-

1. It was originally attached to the back of the wrist, but was moved to the back of the hand to reduce the compounding error effect due to the wrist flexion and abduction angles in calculating finger location.

tive to the Isotrak source. The light-stripe rangefinder, *Cyber*Glove, and Polhemus device are all controlled by a single Sun workstation (Fig. 3).

Experiments were conducted in the following manner:

1. *Take the range image of the scene right before the execution of the task. This is used to determine the location of the object of interest.*
2. *Perform the task while its intensity image sequence and the* Cyber*Glove and Polhemus readings are being recorded.*
3. *Take the range image of the scene right after the task has been performed.*

The task must involve only one manipulative action (e.g., transferring an object only) because it is not possible to sample range images rapidly with this setup. The light-stripe rangefinder takes about 10 seconds to cast 8 stripe patterns and calculate the range values for a 256x240 image. The intensity images (each of resolution 120x128) taken during the performance of the 1-task are sampled at a rate of about 1.5 Hz for verification purposes.

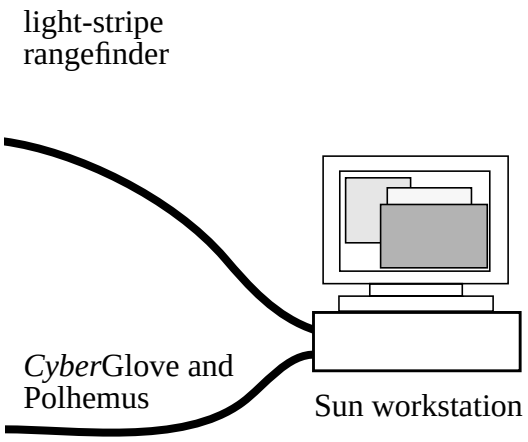


Fig. 3 **Version 1 of data acquisition system**

3.2 Setup Version 2

In the later version of the data acquisition system, we replaced the light-stripe rangefinder with a four-camera active[1] multibaseline system that is connected to the iWarp[2] array via a video interface (Fig. 4). The iWarp system is capable of reading digitized video signals from the four cameras at video rate, i.e., at 30 Hz.

---

1. The word “active” refers to the addition of projected structured lighting during image capture.
2. The iWarp is a high-speed system architecture which is a result of joint effort between Carnegie Mellon University and Intel Corporation [15].

Normally, two cameras are sufficient to recover depth from triangulation. However, having a redundant number of cameras facilitates correct matches between images, which is critical to accurate depth recovery [117]. Details of the video interface are given by Webb, *et al*. [152] while the depth recovery scheme is described at length by Kang, *et al*. [74]. The cameras are verged so that their viewing spaces all intersect at the volume of interest, maximizing camera viewspace utility. In addition, a sinusoidally-varying intensity pattern is projected onto the scene to increase the local discriminability at each image point. The camera calibration procedure (which includes camera adjustment) and the depth recovery scheme are briefly described in Appendix C. Results [74] have indicated that the average errors in fitting planes and cylinders to stereo range data are less than 1 mm at distances of 1.5-3.5 m away from the cameras.

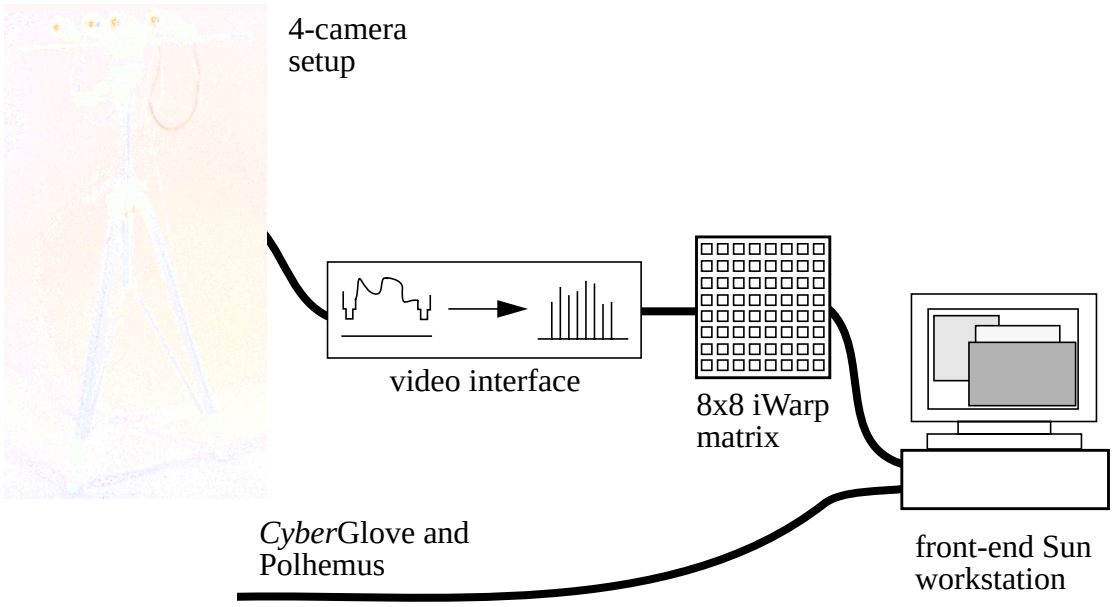


Fig. 4 **Version 2 of data acquisition system**

As before, both the serial line outputs of the *Cyber*Glove and Polhemus device are connected to the host (front-end) Sun workstation. During image capture, the video outputs of the camera system (where all the four cameras are synchronized) are digitized, distributed, and stored in iWarp DRAMs using systolic communication. Once image capture is complete, these images are then chanelled to the front-end workstation to be stored in a storage disk for later depth recovery. We are able to achieve sampling rates of about 4.5-5 Hz.

Chapter 4

Task Recognition Module: Temporal Task Segmentation

The previous chapter describes the function of the observation or data acquisition system, which is to record the perceptual stream of data associated with the operator execution of the task. Once this is done, the next step is to analyze this stream of data in order to extract information about the task for replication. Extracting task descriptions directly from this perceptual stream of data is not straightforward. This is because actions to be identified are local and probably disparate, making global pattern classification techniques such as the Fourier transform generally ineffective.

Instead, the strategy is to preprocess the perceptual data stream by dividing it into general subdivisions. For a task involving grasps, a task can generally be subdivided into three parts: reaching for the object, grasping the object, and manipulating the object (respectively the pregrasp, grasp and manipulation phases, which are described in greater detail later in this chapter).

This chapter describes the temporal segmentation algorithm that partitions a given task sequence into the pregrasp, grasp, and manipulation phases. Temporal task segmentation is important as it serves as a preprocessing step to identify the frames associated with these phases. This information would then be used to focus on the relevant frames in order to characterize the phases in the task. For example, when the grasp phase has been temporally located, the grasp can then be identified using the location of the object and the hand configuration data (Chapter 5). In addition, the motion of the object during each manipulation phase can be extracted.

4.1 Organization of Chapter

This chapter first describes the three different phases in a grasping task. It subsequently reviews some of the characteristics of the pregrasp phase as described in the literature of

human hand movement. The important features that are used in characterizing human hand motion are discussed at greater length. The task segmentation algorithm that uses these features is then detailed; results of the algorithm on several tasks are also shown and discussed.

4.2 Phases in a Grasping Task

As mentioned in the beginning of this chapter, there are three separate identifiable phases in a grasping task:

4.2.1 Pregrasp Phase[1]

This is the first phase of the grasping task which precedes the actual grasp. It is a combination of the trajectory of the hand ([4], [62]) (*hand transportation*), and the temporal changes in finger joint parameters in anticipation of the intended grasp ([4], [62]) (*hand preshape*). The trajectory of the hand is influenced by the distance of the object from the hand ([4], [63]) while the finger joint parameter changes are dependent on the shape of the object ([4], [63], [149]). The hand transportation and hand preshape components have been observed to occur in parallel. Features of the hand preshape such as the approach area, approach volume, and approach axis, as described in [93], can be used to characterize this stage.

4.2.2 Grasp Phase

The pregrasp phase ends and the static grasp phase begins at the moment the hand touches and has a stable hold of the object. The type of grasp employed can be identified at this phase, and can be represented using a grasp abstraction hierarchy described in Chapter 5.

4.2.3 Manipulation Phase

The manipulation phase is characterized by hand motions resulting in the purposeful movement of the object relative to the environment. The grasp is chosen by the operator on the basis of the mobility and dexterity required to manipulate the object.

A manipulative action can be as simple as just translating the object with respect to the environment. It can be as complex as simultaneously transporting (by hand transportation) and precision handling ([81], [88]) the object with the fingertips, changing its pose with respect to both the palm and the environment.

The term of *homogeneous* manipulation to describe the smooth object motion while perturbing a single static grasp is introduced in Perlin, *et al*. [120]. A complete task may comprise several homogeneous manipulations. Perlin, *et al*. [120] propose a structured and hierarchi-

1. This has been variously referred to as reaching ([4], [9], [149]) and target approach ([129], [149]). However, these terms can be easily confused with the hand transportation component of this phase - Jeannerod [62], for example, uses the terms reaching and transportation interchangeably.

cal approach to autonomous manipulation, specifically for the Utah/MIT hand. The scheme involves the establishment of the static grasp taxonomy, from which a library of homogeneous manipulations and subsequently low-level control primitives and sensor interactions may be developed. In our work, we assume homogeneous manipulation (i.e., the same grasp is employed) within a manipulation phase.

4.3 Features for Segmentation of a Task Sequence

In this section, we review research on human hand movement, specifically on the characteristics of reaching motions of the hand (i.e., during the pregrasp phase). Subsequently, we describe the features that are used in our framework of task segmentation. Relevant work on motion representation are also briefly discussed.

4.3.1 Studies in Human Hand Movement

Numerous studies on human hand movement point to commonly established characterizations of the pregrasp phase. The pregrasp phase has been analyzed in terms of two simultaneous activities, namely the hand reaching activity (termed the *hand transportation* component), and the finger activity in anticipation of the grasp (termed the *hand preshape* component[1]) (e.g., [63], [95], [98], [156]).

Typical profiles of the hand transportation speed and grip aperture[2] during the pregrasp phase are shown in Fig. 5. The characteristic inverted bell-shaped curve of the hand transportation speed has been observed by many researchers (e.g., [62], [110]).

Jeannerod ([62], [63]) conducted experiments involving reaching and grasping movements to see how object characteristics affect these movements. His experiments show that object size and orientation affect hand preshape but not hand transportation. However, object distance influences only the hand transportation and not the hand preshape. Another interesting inference from his series of experiments is the temporal coincidence between the starting of the hand preshape aperture reduction and the commencement of the low-velocity phase of the hand tranportation. (The point at which the low-velocity phase begins is at the time where the lowest acceleration occurs.) These happen almost simultaneously after about 75% of movement time had elapsed. Fig. 6 shows the temporal divisions of the hand preshape and hand transportation components into different subphases whose boundaries coincides.

---

1. This has also been referred to as the grasping or manipulation component. The alternative terms are not used here to avoid confusion with the static grasp phase and the manipulation phase of the grasping task.
2. The grip aperture in this context is defined as the separation between the tips of the thumb and index finger.

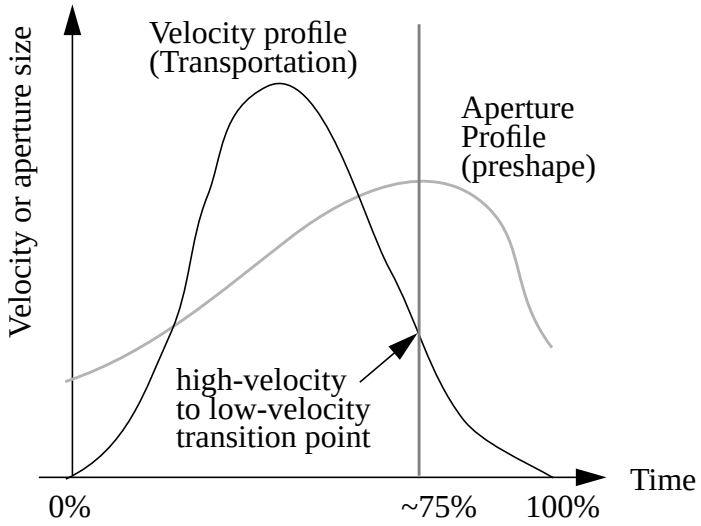


Fig. 5 **Typical pregrasp component profiles**

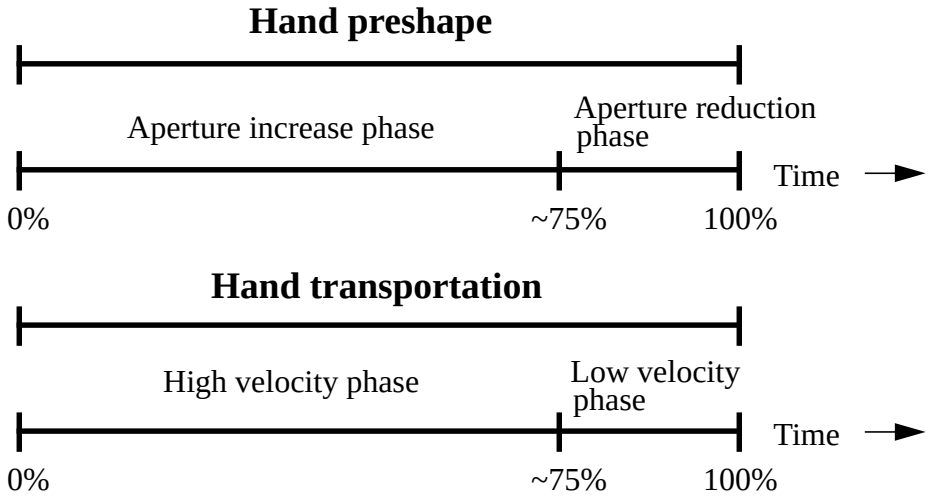


Fig. 6 **Pregrasp phase components**

The hand preshape component is controlled by the distal muscles of the body and appears to be activated by intrinsic or object-centered properties such as object shape and size. In contrast, the hand transportation component is controlled by proximal muscles and appears to be activated by extrinsic or viewer-centered properties such as object location relative to the person [62].

Marteniuk and Athenes [98] found that the maximum grip aperture has very strong linear correlation with the object size and that movement time increased linearly with the decrease in object (disk) size. The latter result is due to the increase in the duration of hand deceleration while the duration of hand acceleration remained constant. The ratio of these two times could perhaps be linked to the precision requirements of hand motion in the task, as Marteniuk, *et al*. [99] suggest. Marteniuk and colleagues [99] noted that the objective of a task,

which dictates the precision requirements of the task, affect trajectory shape. For example, in one of their experiments, the duration of hand deceleration was disproportionately longer for the task of fitting a disk into a well when compared to that for a task of picking up a disk and throwing it into a large box.

Wing, Turton and Fraser [156] report that the grip aperture[1] was greater in cases where reaching movements were performed faster and where there was no visual feedback (i.e., subjects had their eyes closed while reaching for the object).

The results of the research on human hand motion point to the importance of both the grip aperture and speed of the hand in characterizing the pregrasp phase. These studies that highlight the characteristic shapes of the grip aperture and speed profiles indicate that these measures may be used to temporally segment a task sequence into its constituent phases. Both of these metrics (in one form or the other) form the bases of our work on temporal task segmentation. Three quantifiable measures that are proposed are the *hand volume*, the *fingertip polygon*, and the *fingertip polygon normal*.

4.3.2 The Hand Volume, Fingertip Polygon, and Fingertip Polygon Normal

Lyons [93], in describing a conceptual high-level control mechanism for a dexterous hand, defines the following terms:

- approach volume - the volume between the fingers
- approach area - surface formed by joining the fingertips of the preshaped hand by straight lines
- approach axis - outward normal to the approach area through its centroid

Lyons uses these terms in the context of the pregrasp phase. We extend these definitions to the manipulation phase as well; the corresponding terms that we use are:

- hand volume
- fingertip polygon area
- fingertip polygon normal

Fig. 7 depicts the ideas of hand volume, fingertip polygon, and fingertip polygon normal. These features are potentially useful in characterizing a task. In fact, the fingertip polygon is used as one of the primary features for temporal segmentation in our framework.

1. The grip aperture is the separation between the tips of the thumb and index finger.

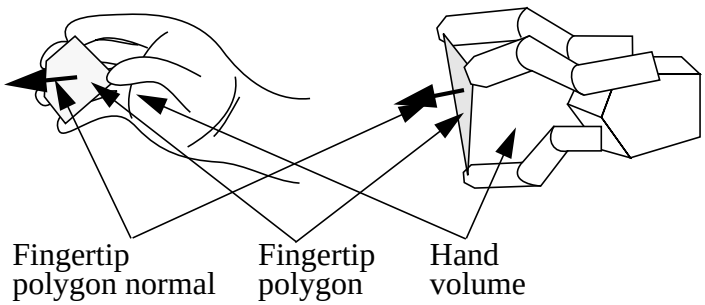


Fig. 7 **Fingertip polygon normal, fingertip polygon, and hand volume for the hand (top) and a manipulator (bottom)**

4.3.3 Calculating the area and centroid of the fingertip polygon

Wing and Fraser [155] found from their experiments that the thumb contributed significantly less in the reduction of grasp aperture that the other fingers. They suggest that the relative stability of the thumb is due to its role in guiding the hand transportation of the pregrasp phase. In light of this research, it would seem reasonable that the position of the centroid within the fingertip polygon would be more heavily influenced by the position of the tip of the thumb.

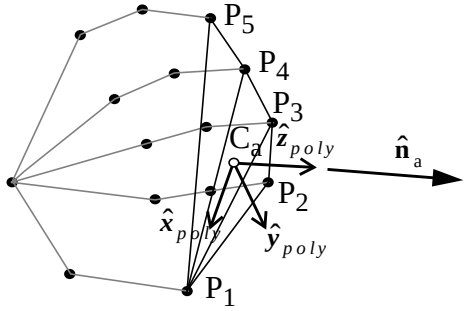


Fig. 8 **Contact web (Chapter 5), the fingertip polygon and the fingertip polygon normal and coordinate frame**

Consider the contact web representation (Chapter 5) of the hand at a given point in time during the pregrasp phase (shown in Fig. 8). The fingertips are denoted as P_1 (tip of the thumb), P_2, P_3, P_4 and P_5; C_a is the centroid of the fingertip polygon while $\hat{\mathbf{n}}_a$ is the fingertip polygon normal. The area of the fingertip polygon is

$$A_{app} = \sum_{k=2}^{4} A_k \tag{1}$$

where A_k is the area of the triangle $P_1P_kP_{k+1}$ (calculated using Heron's formula):

$$A_k = \sqrt{s_k (s_k - l_k)(s_k - l_{k+1})(s_k - m_k)} \tag{2}$$

where

$$s_k = \frac{1}{2}(l_k + l_{k+1} + m_k) \tag{3}$$

l_k is the distance between P_1 and P_k, and m_k is the distance between P_k and P_{k+1}.

The centroid of the fingertip polygon is

$$C_a = \frac{1}{4}\sum_{i=2}^{5}\frac{P_1 + P_i}{2} = \frac{\sum_{i=1}^{5}\omega_i P_i}{\sum_{i=1}^{5}\omega_i} \tag{4}$$

where $\omega_1 = 4$ and $\omega_i = 1$ for $i = 2, ..., 5$.

The fingertip polygon normal is taken to be the normal of the best fit plane to the fingertips (away from the hand). The frame origin of the fingertip polygon is at its centroid with the x-axis defined to be the unit vector pointing towards the thumb fingertip and the z-axis defined as the normal (Fig. 8).

A very important question arises: when do we know that the object has been grasped and at which point does the object move with the hand? One possibility is to use explicit motion boundaries.

4.3.4 Motion Representation using Explicit Boundaries

Rubin and Richards [130] propose to characterize visual motion using explicit boundaries that they define as starts, stops and force discontinuities (step and impulse). When one of these boundaries occurs in a motion, human observers have the subjective impression that some fleeting, significant event has occurred. Iba [50] augments these elementary boundaries with zero crossings in accelerations. While these motion boundaries show promise in task segmentation, they are prone to noise and are less reliable when the sampling rate is low, as was the case in our experimental setup.

4.4 Temporal Segmentation of a Task Sequence

In this section, we first define the notions of a *subtask* and an *N-task*, and describe how a task can conveniently be pictorially represented as a state transition diagram. We then describe the task segmentation algorithm. Subsequently, we show, with examples, how the identified task breakpoints that separate the different phases can be used to identify the grasp and extract object motion in the manipulation phase. Determining object motion in the

manipulation phase is useful in identifying the actions performed on the object during the task.

4.4.1 State Transition Representation of Tasks and Subtasks

An assembly task may comprise a variety of operations such as moving towards an object, picking up an object, moving the grasped object, and inserting one object onto another. It is convenient to represent the task as a series of states and transitions. As mentioned earlier, a task unit (called a *subtask* from this point on) is composed of three phases, namely the pre-grasp, grasp, and manipulation phases. The grasp phase is treated as a transition from the pregrasp phase to the manipulation phase. The state transition diagram for a subtask is shown in Fig. 9(a). A task would minimally comprise these three phases and an ungrasp and depart motions. The ungrasp motion corresponds to the release of the manipulated object. The depart phase is a specialized instance of the pregrasp phase that signals the termination of the entire task sequence. Hence a task has minimally one embedded subtask; such a minimal task is referred to as a *1-task* (Fig. 9(b)). A task with *N* subtasks is termed an *N-task*. The state transition diagram for a general task is depicted in Fig. 10.

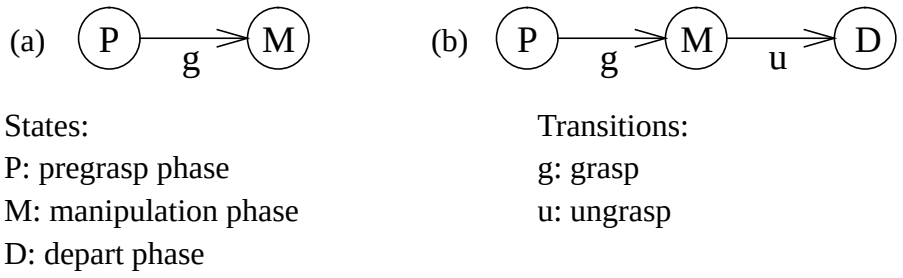


Fig. 9 **State transition diagram of a (a) subtask and (b) 1-task**

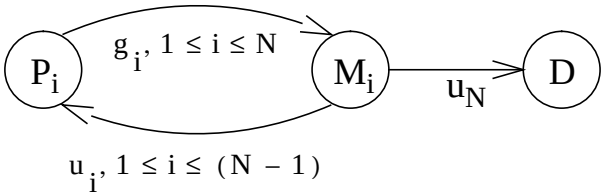


Fig. 10 **State transition diagram of a general task (N-task)**

The state transition diagram representation lays the groundwork for the temporal division of a task sequence and facilitates the visualization of the extracted task components. Note that at this point, this representation is primarily for human visualization and understanding (as opposed to direct system use in [22], [105] and [136], for example). The system does, however, know the relationship between the phases once they have been located, and thus uses the representation in a rudimentary fashion.

4.4.2 Temporal Segmentation of Task into Subtasks

We can segment the entire task into meaningful subparts (such as the different states and transitions described in the previous section) by analyzing both the fingertip polygon area and the speed of the hand. The fingertip polygon area is an indication of the hand preshape while the speed of the hand is an indication of the the hand transportation. While a viable alternative appears to the grip aperture, i.e., the distance between the thumb and the index finger, this feature is more prone to sensor error.

Intuitively, in the pregrasp phase, as a person moves his/her hand towards an object with the goal of picking it up, he/she unconsciously increases the spacing between the fingers in anticipation of the grasp. This yields the characteristic inverted bell curve profile of the fingertip polygon area during this time. The speed profile of the hand also assumes this trend, due to the initial acceleration and the subsequent deceleration. Once the object has been picked and is being moved, we arrive at the manipulation phase of the task. Assuming homogeneous manipulation, the fingertip polygon area remains approximately constant. Once again the speed profile of the hand assumes the inverted bell curve of acceleration and deceleration. By taking into consideration both the fingertip polygon and speed profiles in the pregrasp and manipulation phases, we can more reliably divide the entire task into the following actions: reach for object, grasp object, move or manipulate object, and place object. The breakpoints can be extracted more reliably in this manner than from just the speed or fingertip polygon profile alone. This can be seen by considering the speed and fingertip polygon profiles in Fig. 16 which possess many local minima. The significant values of the fingertip area during the first manipulation phase (frames 11-17) also complicates the segmentation process.

A useful profile to analyze is the profile of the product of the speed and fingertip polygon area at each frame called the *volume sweep rate* profile. The physical interpretation of the volume sweep rate is illustrated in Fig. 11. It measures the motion of the fingertip polygon area due to hand speed in 3-D space. While the hand reaches to grasp the object, both the speed and fingertip polygon area profiles are bell-shaped, and although the peaks are not coincident, its volume sweep rate has an accentuated peak. Thus the volume sweep rate profile of pregrasp phase has a comparatively higher peak value than that of the manipulation phase. This can be seen from the graph based on typical real data in Fig. 12. Meanwhile, during object manipulation, the fingertip polygon area is always less than the peak fingertip polygon area of the reach-object phase just prior to the grasp. This results in a smaller peak value of its volume sweep rate. The volume sweep rate profile is used to get rid of local extrema points (minima) in the speed profile that are not task breakpoints.

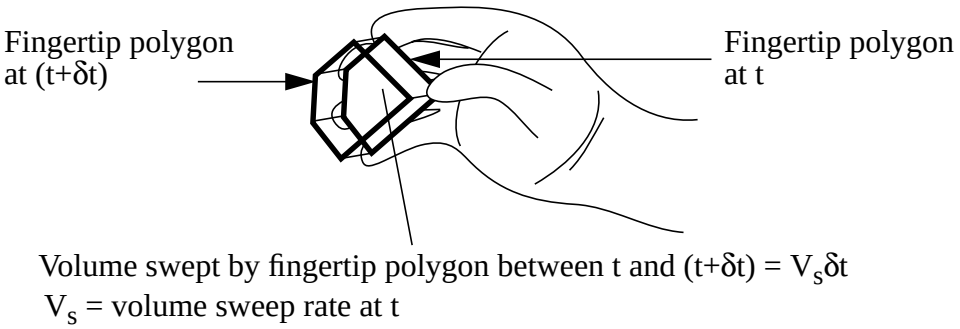


Fig. 11 **Physical interpretation of volume sweep rate**

Our main argument for the use of the volume sweep rate is the accentuation of the peak during the pregrasp phase, which helps to determine the task breakpoints. Looking at the speed curve is not sufficient for two reasons: (1) Coarse sampling and non error-free data which result in actual rests between phases not showing up as zeroes, causing task segmentation to be brittle; and (2) inability to distinguish between actual breakpoints and momentary rests during operations such as turning a screw. Repeated experiments have shown that using the volume sweep rate yielded significantly more reliable segmentation results.

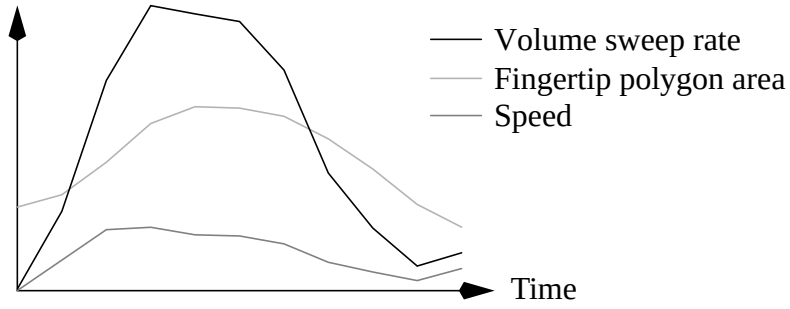


Fig. 12 **Typical profiles of fingertip polygon area, hand speed, and volume sweep rate during the pregrasp phase**

The task breakpoints are a subset of both the speed and volume sweep rate local minima. The algorithm to segment a task sequence into meaningful subsections starts with a list of breakpoints comprising local minima in the speed profile rather than those in the volume sweep rate. This is because while the volume sweep rate is useful in globally locating the true breakpoints, it usually contains more local minima which are not task breakpoints. We can further reduce the size of the set of possible breakpoints by imposing the condition that the speed local minima must also be local minima (within a certain small neighborhood) of the volume sweep rate profile. In a few cases, this extra step turns out to be critical in rejecting certain spurious speed local minima which are the result of momentary pauses during

the pregrasp (reaching out) phase that create problems for the segmentation algorithm. In short, the initial hypothesized set of task breakpoints is given by

$$H_0 = L(S) \cap L_{exp,\delta}(VSR)$$

where *S* is the speed profile, *VSR* is the volume sweep rate profile, *L(M)* is the set of abscissa components of local minima of motion profile *M*, and $L_{exp,\delta}(M)$ is the δ-expanded set of *L(M)* defined by

$$L_{exp,\delta}(M) = \bigcup_{t_i \in L(M)} [t_i - \delta, t_i + \delta]$$

The initial set of task breakpoint hypothesis is based on the observation that the hand and fingers are stationary momentarily during the grasp. The expanded set is to cater for latency between hand and finger motion pauses both during data acquisition and actual execution.

The algorithm to segment a task sequence into meaningful subsections starts with a list of breakpoints comprising local minima in the speed profile. The initial breakpoints are extracted from the speed profile rather than the volume sweep rate profile as while the latter is useful in globally locating the true breakpoints, it contains more local minima. The global segmentation procedure makes use of:

1. *The condition that the pregrasp phases and the manipulation phases interleave;*
2. *The condition that the peak of the volume sweep rate in the manipulation phase is smaller than those of the two adjacent pregrasp phases;*
3. *The condition that the mean of the volume sweep rate in the pregrasp phase is larger than those of the two adjacent manipulation phases; and*
4. *The goodness of fit of the volume sweep rate profiles in the pregrasp phases to gaussian curves*[1].

Let

I_i = interval between breakpoints *i* and *i*+1;

N_I = number of hypothesized intervals;

N_M = number of hypothesized manipulation phases = $\left\lfloor \frac{N_I + 1}{2} \right\rfloor - 1$;

$M_{VSR,i}$ = mean volume sweep rate in I_i;

$M_{APA,i}$ = mean fingertip polygon area in I_i;

1. The profiles were initially fit using parabolas, but the gaussian curve provides a more reliable fit, especially at the sides of the profiles.

$F_{VSR,i}$ = root of sum of squared error in gaussian curve fitting the volume sweep rate profile in I_i;

$F_{APA,i}$ = root of sum of squared error in gaussian curve fitting the fingertip polygon area profile in I_i; and

$$D_i = \begin{cases} F_{VSR,1} F_{VSR,1} \dfrac{M_{APA,1}}{M_{VSR,1}} & , i = 0 \\ \dfrac{1}{2}\left(F_{VSR,2i-1} F_{APA,2i-1} + F_{VSR,2i+1} F_{APA,2i+1}\right) & , 1 \le i \le N_M \\ F_{VSR,N_I} F_{VSR,N_I} \dfrac{M_{APA,N_I}}{M_{VSR,N_I}} & , i = N_M + 1 \end{cases}$$

D_i essentially yields the weighted sum of the RMS errors of gaussian curve fitting of the pregrasp profiles adjacent to the hypothesized manipulation phase. The weight is taken to be the mean polygon area in the pregrasp phase. The objective function associated with the list of breakpoints is given by the mean

$$E = \frac{1}{N_M + 2} \sum_{i=0}^{N_M + 1} D_i$$

The desired breakpoints are obtained by minimizing E. Using the volume sweep rate profile rather than the speed profile reduces the tendency to incorrectly group adjacent phases, since the more pronounced peaks in the pregrasp phase make them much more difficult to erroneously classified together with the adjacent manipulation phases.

Given a tentative list of breakpoints, the algorithm tries all the combinations subject to items 1 to 4 above. Many possibilities are pruned by the conditions in items 2, 3 and 4 above; the combination which yields the best overall gaussian curve fit is then deemed to be the desired task breakpoints.

4.4.3 Experimental Results

Experimental Results using Version 1 of Data Acquisition System

The temporal task segmentation algorithm has been successfully applied to several task sequences, and the results of two of the sequences can be seen in Fig. 15 and Fig. 16. (Note that each motion profile is normalized such that its maximum is 1.) As can be seen from the volume sweep rate profiles, the pregrasp phase peaks are accentuated, thus facilitating phase idenfication. The breakdown and classification of the frames are shown in Fig. 13 and Table 1 (task sequence 1), and Fig. 14 and Table 2 (task sequence 2). Task sequence 1 (a 3-task)

involves only pick and place actions while task sequence 2 (a 4-task) involves pick and place, insert, and screwing actions. For both sequences, the algorithm has correctly identified the frames where an object is grasped and placed, regardless of whether the object is picked and placed, inserted into another object, or screwed into another object.

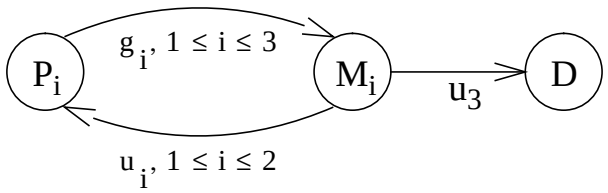


Fig. 13 **State transition representation of task sequence 1 (3-task)**

Table 1 Classification of frames in task sequence 1

i	P_i	g_i	M_i	u_i	D
1	{0, ..., 8}	{9}	{10, ..., 13}	{14}	
2	{15, ..., 18}	{19}	{20, ..., 25}	{26}	
3	{27, ..., 30}	{31}	{32, ..., 37}	{38}	
					{39, ..., 44}

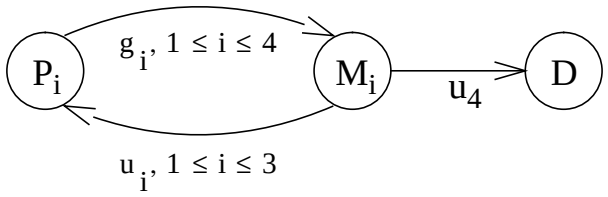


Fig. 14 **State transition representation of task sequence 2 (4-task)**

Table 2 Classification of frames in task sequence 2

i	P_i	g_i	M_i	u_i	D
1	{0, ..., 9}	{10}	{11, ..., 17}	{18}	
2	{19, ..., 23}	{24}	{25, ..., 35}	{36}	
3	{37, ..., 43}	{44}	{45, ..., 51}	{52}	
4	{53, ..., 58}	{59}	{60, ..., 90}	{91}	
5					{92, ..., 101}

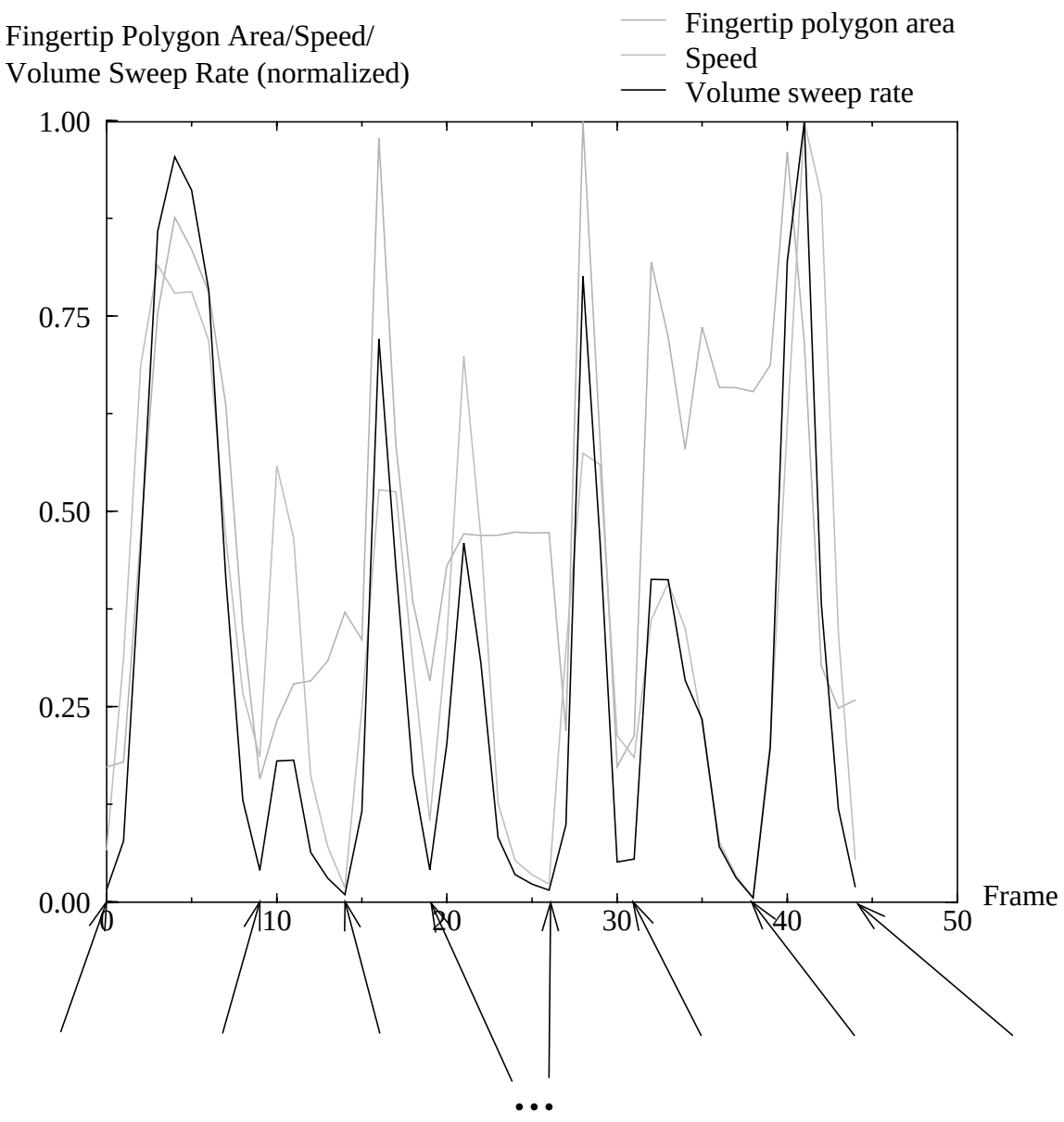


Fig. 15 **Identified breakpoints in task sequence 1 (a 3-task).**

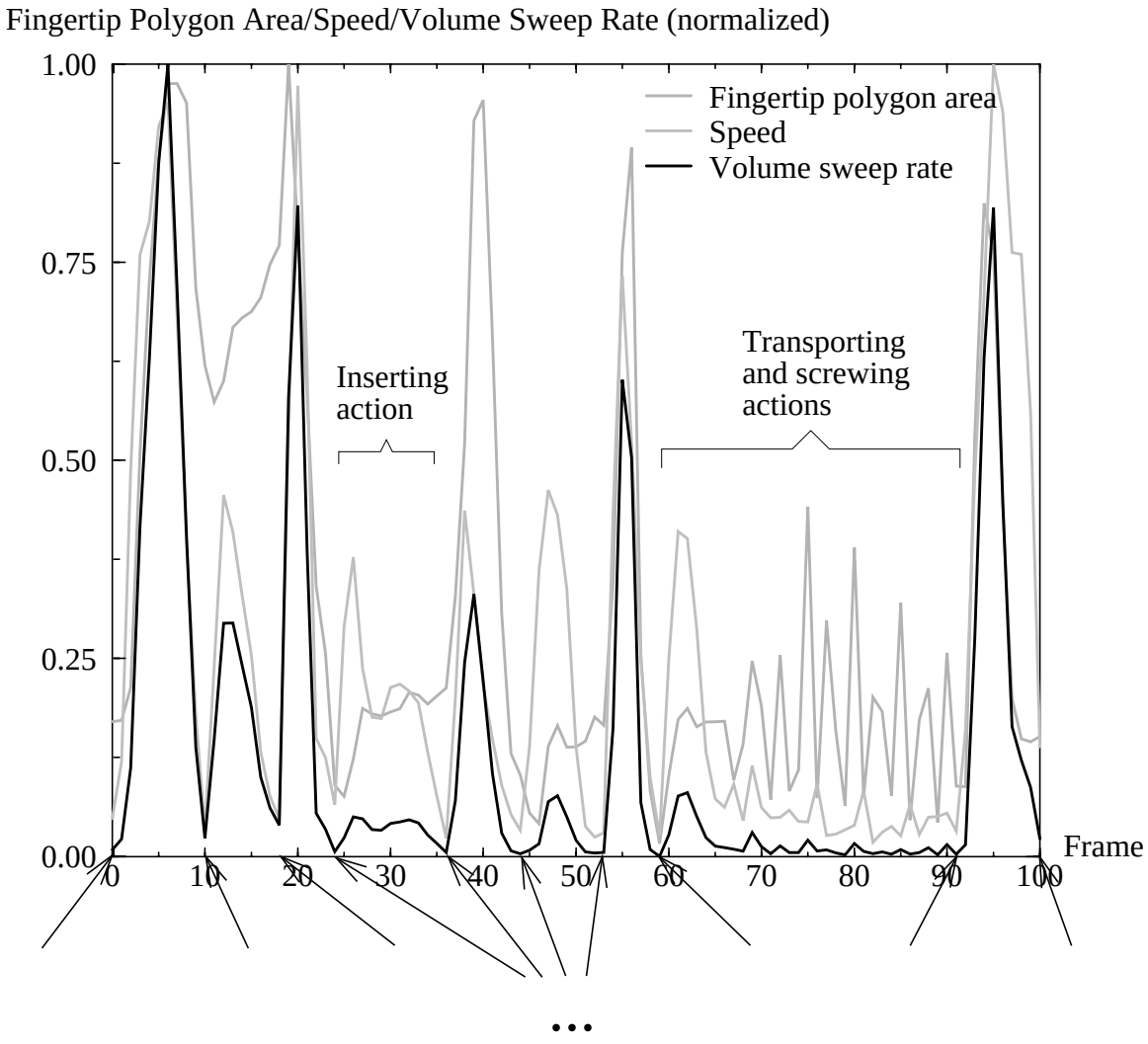


Fig. 16 **Identified breakpoints in task sequence 2 (a 4-task).**

Experimental Results using Version 2 of Data Acquisition System

Two examples of task sequences recorded using the four-camera multibaseline system, *Cyber*Glove, and Polhemus device are shown in Fig. 17 (task sequence 3) and Fig. 18 (task sequence 4). Task sequence 3 involves two subtasks: transporting a screw and turning it into a threaded hole, and inserting a cylinder into a hole. Task sequence 4 involves two pick-and-place actions on a cylinder using a power grasp and a box using a precision grasp.

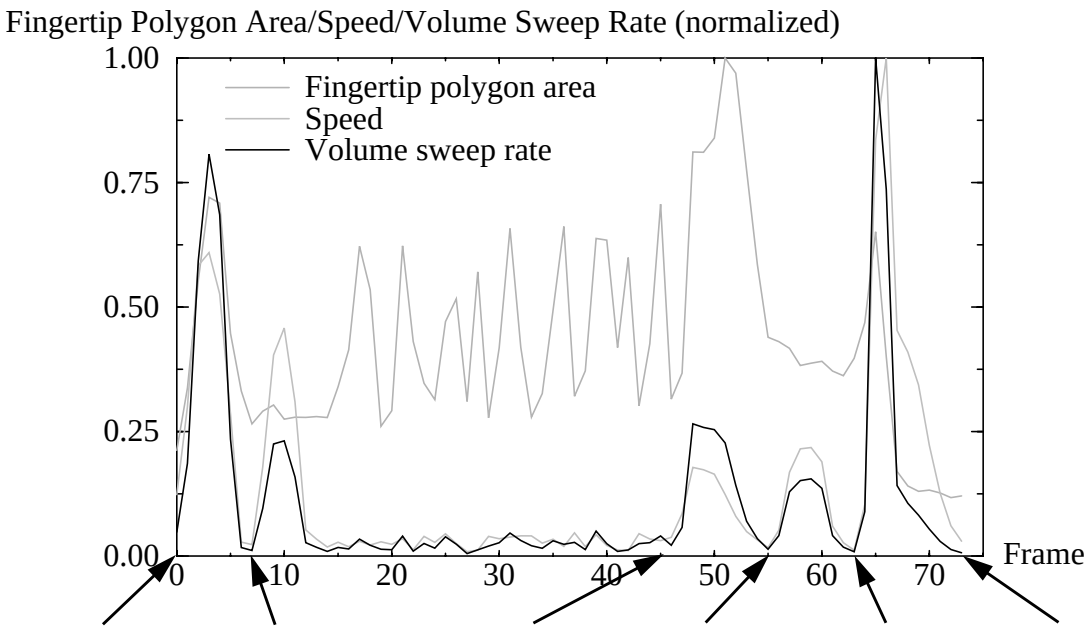


Fig. 17 **Identified breakpoints in task sequence 3 (a 2-task)**

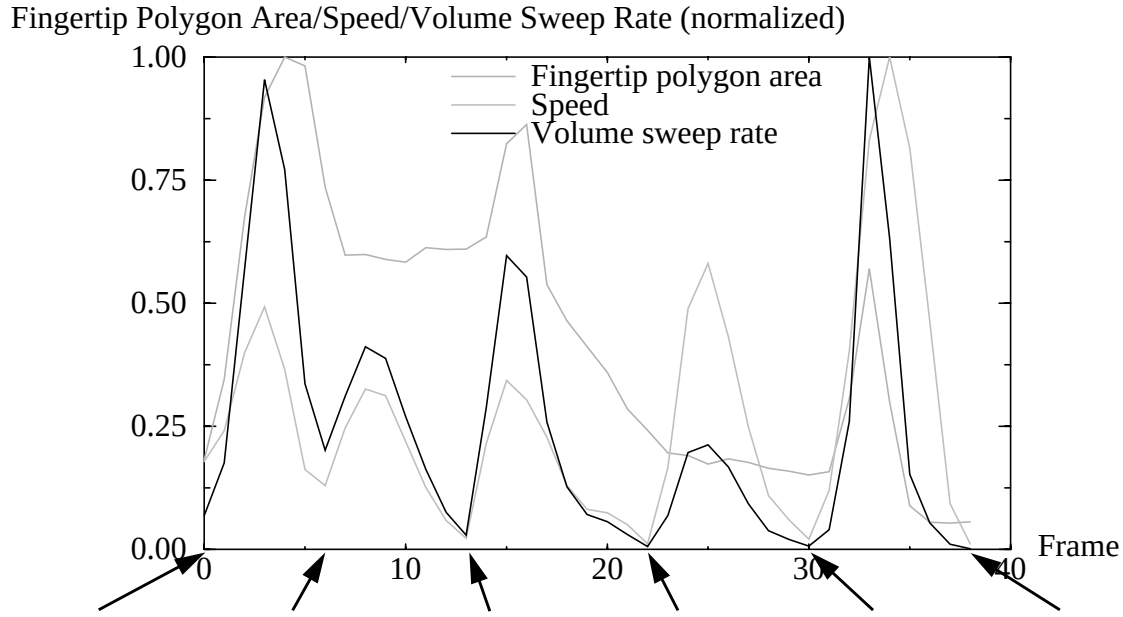


Fig. 18 **Identified breakpoints in task sequence 4 (a 2-task)**

4.4.4 Incorrect Segmentation

Incorrect segmentation occurs when hand motion is too rapid for an effective number of samples to be taken or the operator deliberately misleads the system by slowing down or narrowing the grip aperture at an unappropriate time during the pregrasp phase. A typical example of deliberately misleading the system is shown in Fig. 19. This task sequence is actually similar to that in Fig. 18, with the same setup. However, in this case, the operator deliberately narrows the aperture grip in the middle of the pregrasp phase (see the circled area in Fig. 19). The assumption of the characteristic bell curve associated with the pregrasp phase is thus violated, causing erroneous breakpoint identification.

The problem of incorrect segmentation may be avoided by checking hand proximity to objects using range data, but this is done at a very high computational cost. There is also a problem of setting proximity thresholds. This means of verification has not been implemented here.

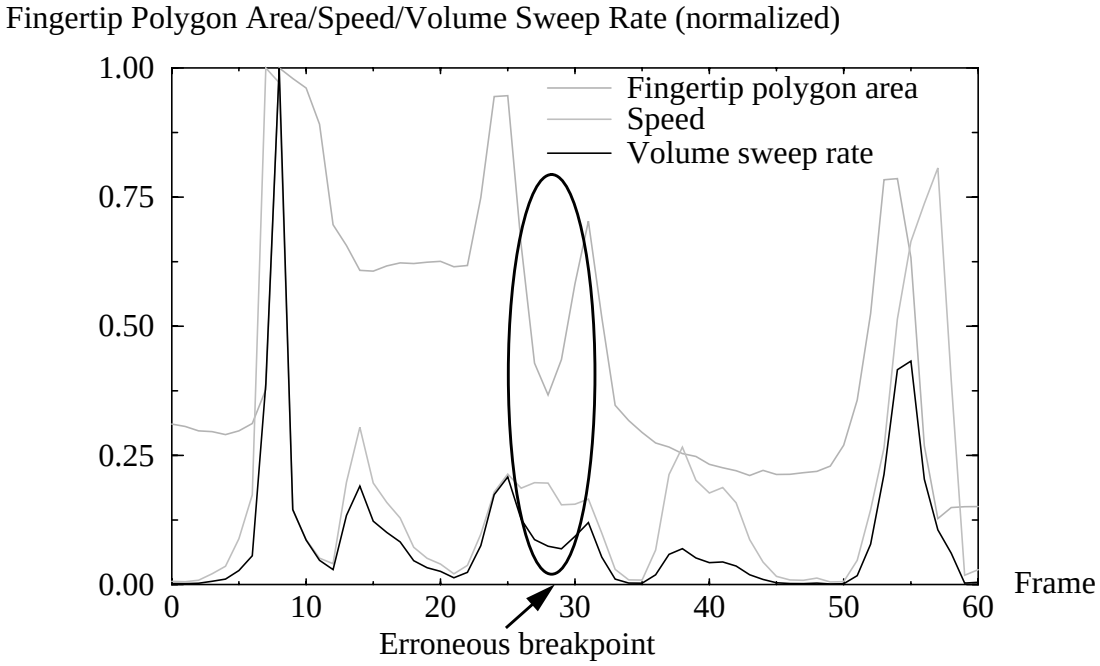


Fig. 19 **Example of incorrectly identified breakpoint**

4.5 Summary of Chapter

A task comprises three identifiable phases, namely, the pregrasp, grasp, and manipulation phases. By using the motion profiles of the task, we show that the task can be automatically temporally segmented into these phases. The motion profiles are those of the fingertip polygon area (area of polygon whose vertices are the fingertips) and the speed of the hand motion. We introduce the notion of the *volume sweep rate*, which is the product of the fin-

gertip polygon area and the hand speed. The volume sweep rate profile is also used in the task division algorithm. The successful application of this algorithm on real task examples demonstrates its viability. The temporal task segmentation process is important as it serves as a preprocessing step to the characterization of the task phases. Once the breakpoints have been identified, steps to recognize the grasp and extract the object motion can then be carried out. The following chapter describes how the human grasp can be identified.

Chapter 5

Task Recognition Module: Human Grasp Recognition

Once the task sequence has been temporally segmented as described in the previous chapter, the first thing that can be done is human grasp identication. Not only is the resulting grasp description used as a guide for manipulator grasp planning, but it also allows high-level operator verification. This chapter describes how the human grasp is identified from perceptual data, and what the grasp representation used to achieve the recognition.

5.1 Organization of Chapter

This chapter begins by describing the human model used in our work. In order to recognize a human grasp from perceptual data, we use a human grasp representation (called the *contact web*) and a grasp taxonomy based on this grasp representation. The grasp recognition scheme that uses this taxonomy is then described. Once grasp recognition has been accomplished, a grasp abstraction hierarchy that incorporates different levels of descriptions can subsequently be constructed.

5.2 Hand Model

An articulated hand model whose finger movements closely resemble those of an actual hand is used to infer the grasp observed in the scene. The human hand is depicted in Fig. 20.

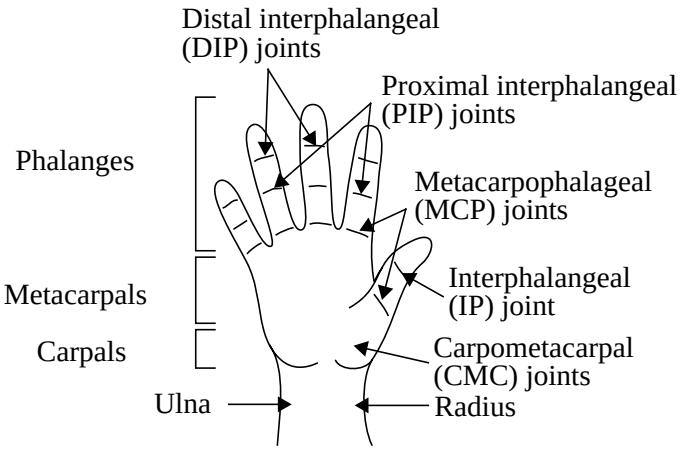


Fig. 20 **Bones and joints of the human hand (adapted from Iberall and MacKenzie [56]).**

Each finger is approximated to have four degrees of freedom. The four angular parameters associated with finger movement (except the thumb) are directly associated with the degree of finger abduction and the degrees of flexion at the metacarpophalangeal, proximal interphalangeal and distal interphalangeal joints (Fig. 20). For the thumb, they are the angular flexions in the carpometacarpal joints (two parameters), the metacarpophalangeal joint and the interphalangeal joint. In this hand model, limits of these angular parameters which are consistent with anatomical and physiological studies of the hand (e.g., [3], [64], and [147]) are imposed. Flexion angles are defined with respect to the hand frontal plane[1] while the abduction angles are defined with respect to the sagittal planes[2].

In our preliminary work on human grasp recognition, the human hand finger segments are modeled by cylinders. Buchholz and Armstrong [20] model hand segments with ellipsoids for ease of analytic determination of contact points between the hand and held object, which is also modeled by an ellipsoid. The ellipsoid-ellipsoid contact algorithm allows modelling of soft tissue penetration. While the ellipsoidal modelling is suitable for simulation, it is less suitable for tracking.

5.3 Classification of Grasps

There has been a lot of study in the medical community on the grasping capabilities of the human hand, from the anatomical and functional points of view. Schlesinger [133] and Taylor and Schwarz [147] associate human grasps primarily with the object shape in their categorization of six grasps (cylindrical, fingertip, hook, palmar, spherical and lateral). Griffiths' [40] grasp classification is also based on objects of varying form. He partitions the functions of the hand into cylinder grasp, ball grasp, ring grasp, pincer grasp and plier grasp. McBride

1. The frontal plane is the plane parallel to a flat hand with fingers extended.
2. The reference sagittal plane of a finger is the plane perpendicular to the frontal plane passing through the principal long axis of the fully adducted and extended finger.

[104] took a different approach in dividing the function of the hand: his classification depends on the parts of the hand which participate in the grasp (grasp with the whole hand, grasp with thumb and fingers, and grasp with finger and palm). These classifications, while expressive and intuitively informative, do not reflect a fundamental analysis of the hand as an entity. They are either too dependent on the shape of the held object ([40], [133], [147]), or arbitrary without any particular functional basis ([104]).

Napier [112], on the other hand, dichotomized grasps into precision grasps and power grasps[1]. His classification of grasps is based on the purpose of the task, shape and size of the object, and the posture of the fingers. This division of grasps into precision and power grasps is the most widely accepted and used by researchers in the medical, biomechanical and robotic fields. A power grasp is used for higher stability and security at the expense of object maneuverability, while the converse is true for a precision grasp. A precision grasp is characterized by a small degree of contact between the hand and the object. In this type of grasp, the object is normally pinched between the thumb and the flexor aspects of at least one finger. In a power grasp, however, the object is held tight by the fingers and the palm[2]. The major classifications of a power grasp are the cylindrical power grasp and the spherical power grasp. In a cylindrical power grasp, the thumb can either be adducted for some element of precision, or abducted for more clamping action on the object. Henceforth the cylindrical power grasp refers to the former type while the “coal-hammer” grasp refers to the latter type. Sollerman [138] uses a different terminology for power grasps; he refers to the cylindrical power grasp, “coal-hammer” grasp, and spherical power grasp as the diagonal volar grasp, transverse volar grasp, and spherical volar grasp respectively.

Cutkosky and Wright [32] construct a hierarchical tree of grasps beginning with Napier’s distinction between precision and power grasps. At the lowest level, a grasp is chosen based on object geometric details and task requirements. However, not only is the taxonomy incomplete, but because the grasp classification is discrete, there may exist problems in categorizing grasps in intermediate cases (e.g., the shape of the object is somewhere between being strictly prismatic and strictly spherical). In these cases, determination of the type of grasp will then be dependent mostly on human judgment rather than on reasoning.

In our effort to automate the recognition of grasps, we require a grasp taxonomy which could provide a systematic way of identifying grasps based on the hand configuration and object shape. Cutkosky and Wright’s grasp taxonomy is not suitable for use in our work because, in addition to its limitations mentioned above, it presumes *a priori* knowledge of the task requirements which are not available in our problem domain. We propose a grasp

1. He actually referred to them as precision grips and power grips. The term “grasp” is used throughout this document for consistency.
2. An exception is the lateral pinch, which is the grasp employed when turning a key in a lock. This grasp involves the thumb and the radial side of the index finger.

taxonomy based on the analysis of the effective contact points of the hand with the grasped object. The effective contact point of a finger segment represents the surface contact of that segment with the object. The resultant spatial pattern of contact points forms what we call a *contact web*.

5.4 The Contact Web

5.4.1 Definitions

A *contact web* is defined as a 3-D graphical structure connecting the effective points of contact between the hand and the object grasped. When parts of a finger or palm make contact with the grasped object, the actual contact area is finite. A point contact is useful in representing the contact between the phalangeal segments and palm, and the object because of ease of representation and analysis, and accommodation of uncertainty in grasping. The shape and cardinality of the contact web yield important information about the type of grasp used.

Intradigital contact points are contact points along the same finger. *Interdigital contact points* are those located at different fingers. The contact notation adopted is illustrated in Fig. 21.

P_I^i is the intradigital contact point set for the i^{th} finger (i = 0 (the thumb), 1, 2, 3, 4);

E.g., $P_I^1 = \{C_{13}\}$ refers to the finger tip contact point set of the index finger.

P_C = {fingers in contact with object};

$P_H = \{ P_I^i : i \in P_C \}$ (the contact point set);

$N_0(P_H)$ = cardinality of P_H = number of fingers in contact with object;

$N_1(P_H) = N_0\left(\bigcup_{i \in P_C} P_I^i \right)$ = total number of contact points.

Note: $N_1(P_H)\big|_{max} = 15$.

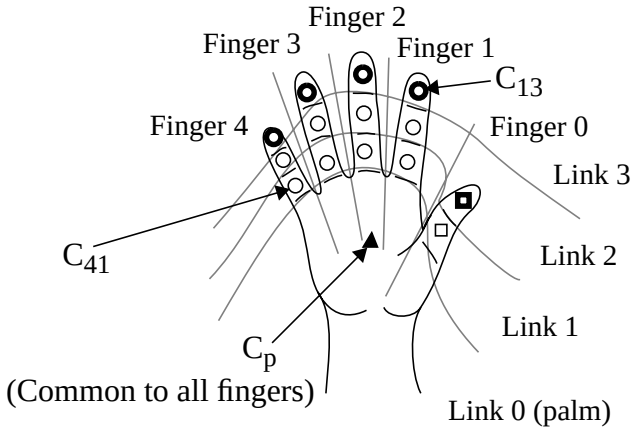


Fig. 21 **Contact Notation on the right hand (palmar side)**

5.4.2 A Taxonomy based on the Contact Web

Cutkosky and Wrights' taxonomy [32] provides a systematic way of finding a grasp that will satisfy a particular set of task requirements and a particular object shape. However, such a taxonomy is difficult to use if (as in our case) the task requirements are not known *a priori*. We propose a grasp taxonomy which is based on the contact web. It provides a systematic way of recognizing grasps from the hand configuration and the object shape. In addition, it provides a more continuous classification of grasps by not restricting the classification to discrete grasp groups.

Fig. 22 shows the proposed major classifications of grasps and the type of contact web associated with each major grasp group. The contact web provides an easy, compact and convenient framework for describing grasps, giving insights into the type and shape of the object grasped, and possibly, the tasks involved. Note that the contact point of the palm is merely an artifact which represents the effective contact of the palm with the grasped object. This point may or may not be physically in contact with the object.

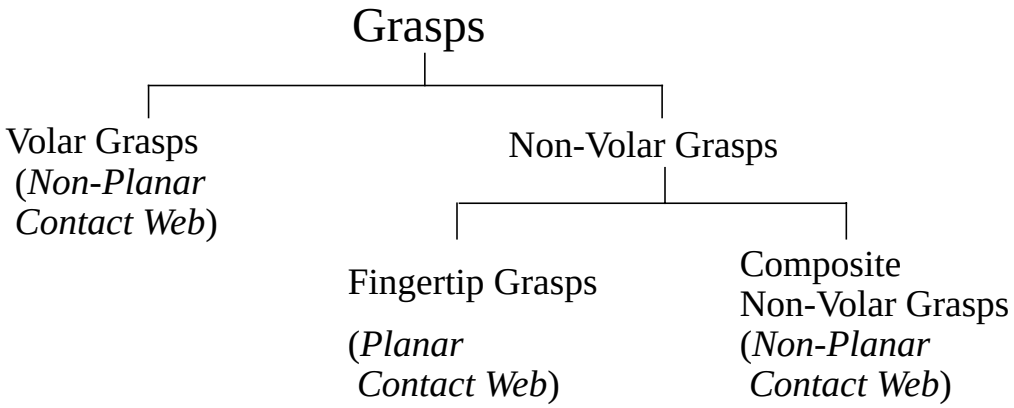


Fig. 22 **Major classifications of grasps for recognition**

A power grasp is characterized by a high degree of contact between the hand and the held object. This allows high clamping forces on the object. A feature that we use to first distinguish between grasps is the involvement of the palm surface in the grasp. Grasps which involve the palm surface are called *volar grasps* while others are called *non-volar grasps*. All volar grasps are power grasps. All but one type of non-volar grasps are precision grasps[1]. The exception mentioned is the lateral pinch, which is the grasp assumed by the hand when turning a key in a lock. Although the lateral pinch does not involve the palm surface, it emphasizes on the security of the object rather than its dexterity; hence it is a power grasp. This is the reason why volar and non-volar grasps are not equivalent to power and precision grasps respectively, and are not labeled as such in our taxonomy. The non-volar grasps are further classified as *fingertip grasps* and *composite non-volar grasps*. The fingertip grasp involves only the fingertips while the composite non-volar grasp involves surfaces of other segments of the fingers in addition to the fingertips. The major grasp classifications are shown in Fig. 22, while the effective contact point notation is depicted in Fig. 21.

One interesting feature of a category of grasps is whether the contact web associated with that category is planar or non-planar. The contact web formed by a volar grasp is spatially non-planar (except for the platform push[2]). In most non-volar grasps where the areas of contact between the object and the hand are those of the fingertips (fingertip grasps), the associated contact web is approximately planar. However, there are at least two identifiable cases of non-volar grasps where the contact web is non-planar, namely the lateral pinch and the pinch grasp. These are separately grouped as composite non-volar grasps.

The contact web enables a more continuous categorization of grasps as shown in Fig. 23 and Fig. 24. The degree of membership to a strictly prismatic grasp or spherical/disc grasp lies in the degree of fit of the contact points to the respective shapes. In addition, the contact web facilitates a mathematical framework for the recognition of grasps as described in section 5.6. Finally, as discussed in this section, this grasp taxonomy provides a systematic means of grasp discrimination from observation.

5.4.3 Comparisons with Other Grasp Frameworks

Cutkosky discusses how to choose grasps based on task requirements and object shape, and he implemented an expert system called “Grasp-Exp” to do this [29]. Cutkosky and Howe

---

1. Napier [112] regards volar grasps as power grasps and non-volar grasps as precision grasps. This view is shared by various other researchers in the medical and biomechanical fields (e.g., [20], [89]). However, this would not be *strictly* true if we adhere to the position that the type of grasp should be classified according to the predominance of either power or precision in the grasp (which, interestingly, Napier [112] adheres to as well). The demand for power is higher than the demand for precision in the lateral pinch; though it is a non-volar grasp, it is, by the power predominance definition, a power grasp. See text.
2. The platform push is a *non-prehensile* grasp; non-prehensile grasps are not considered in this work.

[30], in a similar vein, relate grasp attributes such as dexterity and precision to analytic measures such as manipulability and isotropy in their grasp analysis. Iberall and MacKenzie [56] concentrate on finding a grasp solution for the controller given anticipated object properties and predictable interaction outcome in terms of opposition space and virtual fingers. Iberall [53] describes a neural network that maps qualitative task and object properties (surface length in terms of finger span, object width, amount of forces, and level of task precision) onto a desired prehensile posture. The mapping is based on empirical evidence.

In each of these cases, an explicit analytical framework is not provided to *identify* a given grasp. In our scenario, object description is available while task description is not. The system has to somehow be able to determine what grasping strategy has been employed based on viewing a multiple sequence of the task.

Nguyen and Stephanou [114] describe a topological algorithm for continuous grasp planning. Their paper proposes a topological algorithm to determine a grasp expressed in terms of low-level joint variables, given a high-level task description. Again, the assumptions made and the domain are different. In our framework, no task description is available. Instead, it is inferred.

Pao and Speeter [118] determine the matrix transformation linking the joint parameters of the human hand and a multifingered robotic hand based on a predefined set of hand poses. The robotic hand posture corresponding to any given hand configuration is then interpolated using this matrix transformation. There is no higher level of abstraction in their method as correspondence between the human and robotic hand configurations is established based on low-level joint angles. As a result, it may be difficult to generalize this scheme to less anthropomorphic robotic hands and for complicated tasks.

Our framework provides a direct mathematical means to use higher-level conceptual terms for describing the physical configuration of the hand. As mentioned earlier, it also provides a more continuous taxonomy of grasps.

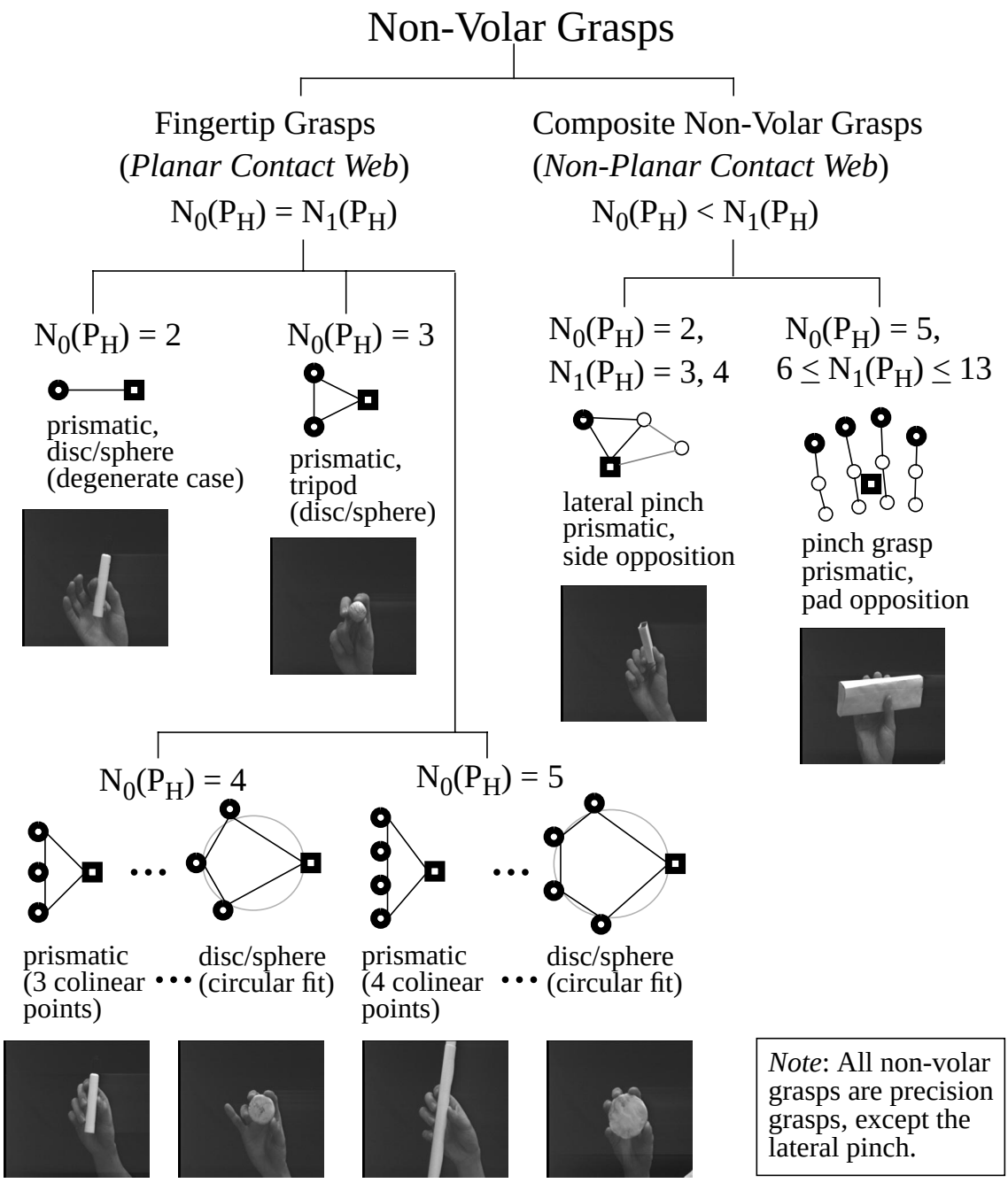


Fig. 23 **Classification of non-volar grasps**

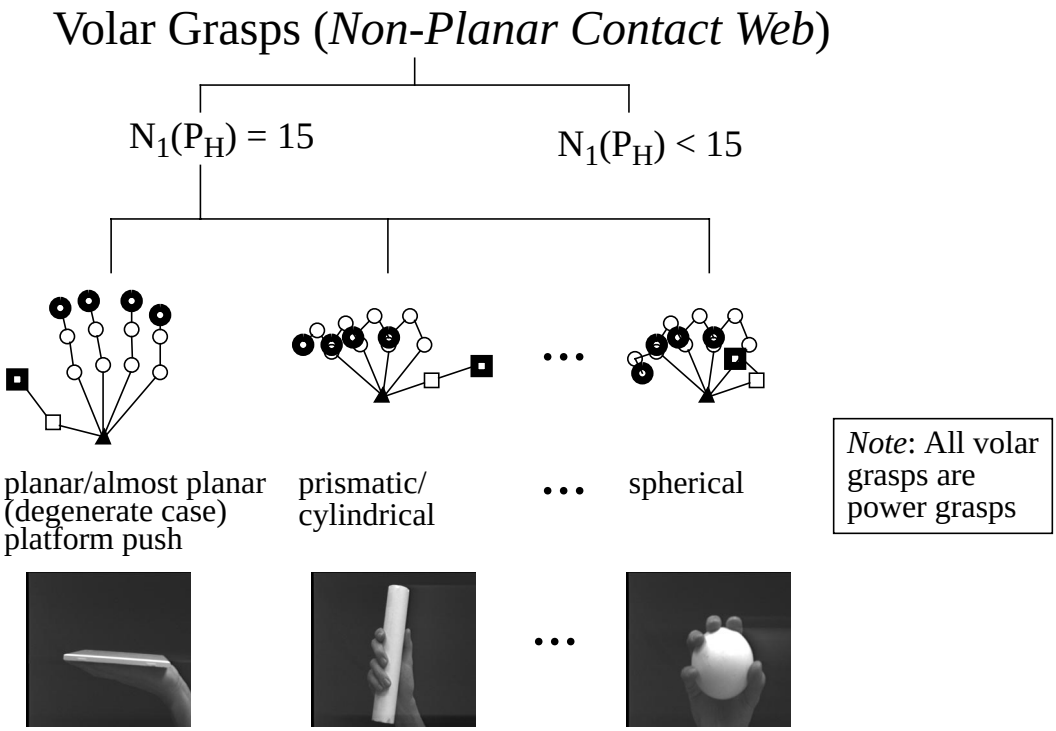


Fig. 24 **Classification of volar grasps**

5.5 Virtual Fingers and Opposition Space

By analyzing the contact web, medium level grasp concepts such as virtual fingers and opposition space, can be described. These two concepts, in turn, are the key elements in characterizing the type of grasp and indicating the functionality of the grasp.

Arbib, *et al*. [4] introduced the concept of the *virtual finger*: a functional unit which comprises at least one real physical finger (which may include the palm). The real fingers composing a virtual finger act in unison to apply an opposing force on the object and against the other virtual fingers in a grasp. This concept replaces the analysis of the mechanical degrees of freedom of individual fingers by the analysis of the functional roles of forces being applied in a grasp.

Cutkosky and Howe [30] suggest that virtual fingers correspond to independently controlled contact sites, and oppositions correspond to internal grasp forces. While Iberall [51] indicates that the precision tripod grasp consists of two virtual fingers (thumb as one and the index and third fingers as the other), Cutkosky and Howe [30] state that it may have either two or three virtual fingers, depending on the amount of coupling between the index and third fingers. We agree with the latter view. Finger force interaction is the basis for virtual finger composition determination in our work.

Iberall, *et al*. [54] define *opposition space* as “the area within the coordinates of the hand where opposing forces can be exerted between virtual finger surfaces in effecting a stable grasp.” They show that prehensile grasps involve combinations of the three basic oppositions shown in Fig. 25. Pad opposition is used in precision grasps, while palm opposition provides a more powerful hold on the object. Finally, side opposition is a compromise between these two oppositions in terms of dexterity and strength of the grasp. Opposition space is an important concept in characterizing grasps.

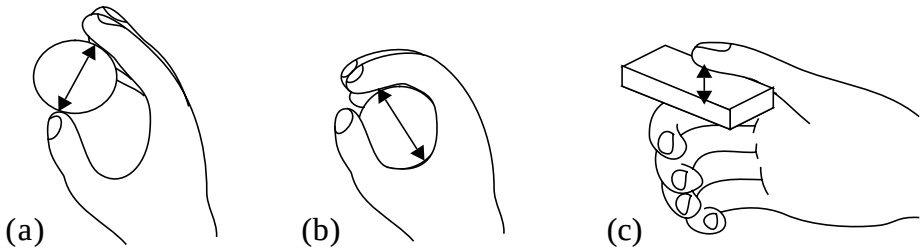


Fig. 25 **Types of opposition (adapted from Iberall, *et al*. [56]). (a) Pad opposition, (b) Palm opposition, (c) Side opposition**

5.6 Recognizing Grasps from the Contact Web

We now illustrate how the contact web can be used to identify the grasp. As mentioned earlier, there are three different types of opposition: pad opposition, palm opposition, and side opposition. We start with the simplest type of opposition, namely, pad opposition, and then proceed to side opposition. The detailed analyses involving these oppositions in the next two subsections constitute the main ideas embodied in the mathematical framework for grasp recognition.

5.6.1 Pad Opposition Only

There are at least two virtual fingers to effect this opposition. The *degree of force coupling* between any two given forces $\mathbf{f}_{i,cpp}$ and $\mathbf{f}_{j,cpp}$ is defined to be their normalized dot product:

$$D_c(i,j) = \frac{\mathbf{f}_{i,cpp} \bullet \mathbf{f}_{j,cpp}}{|\mathbf{f}_{i,cpp}||\mathbf{f}_{j,cpp}|}$$

The following is a proposed analytical method of determining the mapping of all the fingers touching the grasped object into either one, two or three virtual fingers. Note that this method does not presume the mapping of the thumb into one virtual finger.

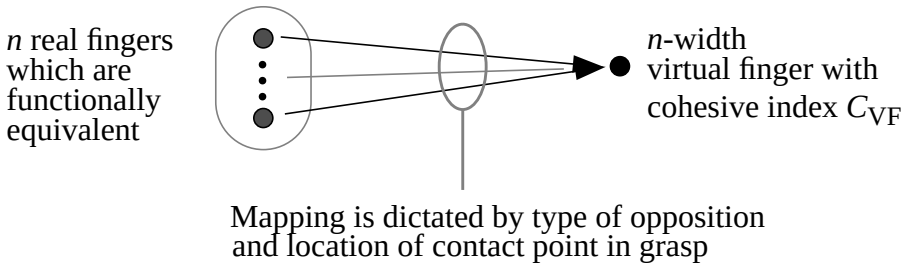


Fig. 26 **Virtual finger mapping under the influence of opposition space and point contact placement**

This method is based on the premise that the mapping and number of virtual fingers depend on the location and degree of coupling between the fingers in contact with the grasped object. This philosophy is illustrated in Fig. 26. It quantifies the degree of coupling between fingers and introduces the concept of the *cohesive index* of a virtual finger. The cohesive index of a virtual finger indicates the degree to which the real fingers mapped into it are functionally equivalent. Let the normal forces on the contact polygon plane (CPP) be denoted as $\mathbf{f}'_{i,cpp}$ (i=1, ..., n), and the actual internal projected forces acting on the object be represented by $\mathbf{f}_{i,cpp}$ (i=1, ..., n). Assume that there are n_{RF} real fingers in contact with the grasped object and that $\mathbf{f}'_{i,cpp}$ are known for i = 1, ..., n_{RF}. Assume also, that $\mathbf{f}_{i,cpp} \approx \mathbf{f}'_{i,cpp}$. The virtual finger membership index between fingers i and j (each with only one contact point for the moment) is defined as:

$$m_{ij} = \frac{\min\left(|\mathbf{f}_{i,cpp}|, |\mathbf{f}_{j,cpp}|\right)}{\max\left(|\mathbf{f}_{i,cpp}|, |\mathbf{f}_{j,cpp}|\right)} \frac{1 + D_c(i,j)}{2}$$

It can be seen that $0 \le m_{ij} \le 1$. Two real fingers are more likely to be members of the same virtual finger if the force vectors at the contact points are similar (i.e., both in terms of force direction and magnitude). Obviously $m_{ii} = 1$ and $m_{ij} = m_{ji}$. Let VF_k denote the set of real fingers hypothetically belonging to the k^{th} virtual finger. Then the cohesive index for that virtual finger is defined as the geometric mean of all the pairwise virtual membership indices:

$$C_{VF,k} = \prod_{\substack{i,\, j \in VF_k \\ j \ge i}} m_{ij}^{\xi}$$

where

$$\xi = \binom{N(VF_k)}{2}^{-1}$$

$N(VF_k)$ being the number of real fingers in virtual finger VF_k and ξ is the reciprocal of the number of possible pairs of real fingers in virtual finger VF_k. $C_{vf,k}$ characterizes the similar-

ity of action of the real fingers in VF_k. If all the fingers in VF_k act in unison, i.e., exert forces equal in magnitude and direction, then $C_{vf,k} = 1$. However, if any two fingers in VF_k exert forces in opposite directions, then $C_{vf,k} = 0$.

The problem of determining the number of virtual fingers and the constituents of each virtual finger can be described as a non-linear mixed program:

Maximize

$$C_{eff} = \left(\frac{1}{n_{VF}!} \prod_{i=1}^{n_{VF}} C_{VF,i} \right)^{\frac{1}{n_{VF}}}$$

subject to

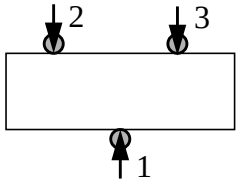
$$n_{VF} \in \{1, 2, 3\}$$

$$\bigcup_{i=1}^{n_{VF}} VF_i = RF$$

$$VF_i \cap VF_j = \varnothing, \qquad i \neq j; \qquad (1 \leq i, j \leq n_{VF})$$

The product term with the exponent in the objective function C_{eff} is the geometric mean of the membership indices of the hypothesized virtual fingers. This ensures that the division of real fingers into virtual fingers is done in such a way that the real fingers in each virtual finger act on the object in as similar a manner as possible. C_{eff} is called the *grasp cohesive index*. The remaining factor in the objective function is designed to favor a smaller number of virtual fingers should there exist equivalency in the objective function (without this factor) for different hand configurations comprising different numbers of virtual fingers. RF is the set of real fingers in contact with the grasped object. For the following examples, for simplicity, it is assumed that all the forces exerted are of unit magnitude (this assumption does not detract from the basic principle). This assumption is altered somewhat for the experiments which are described in subsequent sections.

Example 1 (see Fig. 27)

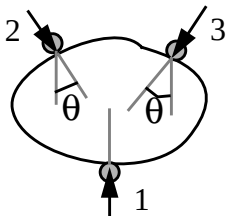


$m_{12} = 0$
$m_{13} = 0$
$m_{23} = 1$

Fig. 27 **Illustration for Example 1**

For the simple case of the thumb and two fingers holding a prismatic object in place, the highest value for C_{eff} is obtained for $VF_1 = \{1\}$ and $VF_2 = \{2, 3\}$. $C_{VF,1} = 1$ and $C_{VF,2} = 1$, giving $C_{eff,max} = 0.707$. For $VF_1 = \{1\}$, $VF_2 = \{2\}$ and $VF_3 = \{3\}$, $C_{eff} = 0.550$.

Example 2(a) and (b) (see Fig. 28)



(a) $\theta = 30°$
$m_{12} = 0.07$
$m_{13} = 0.07$
$m_{23} = 0.75$

(b) $\theta = 50°$
$m_{12} = 0.18$
$m_{13} = 0.18$
$m_{23} = 0.41$

Fig. 28 **Illustration for Example 2(a) and (b)**

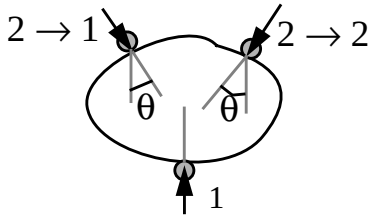
In this example, the object held is no longer prismatic but roughly resembles an ellipse. For case (a), the highest value for C_{eff} is again obtained for $VF_1 = \{1\}$ and $VF_2 = \{2, 3\}$ (here $C_{eff,\,max} = 0.612$). However, in case (b), the highest value for C_{eff} is obtained for $VF_1 = \{1\}$, $VF_2 = \{2\}$, and $VF_3 = \{3\}$ ($C_{eff,\,max} = 0.550$). (Note that θ is the angle measured with respect to the vertical line to which the force direction at contact point 1 is anti-parallel.)

5.6.2 Side Opposition Only

Side opposition involves two fingers – the thumb and the index finger. Contact points that are part of the same finger (i.e., intradigital contact points) are automatically grouped together. This means that either one or two virtual fingers exist in this type of grasp configuration. The k^{th} "composite" finger comprising l (l = 2 or 3) intradigital contact points is denoted by $k \rightarrow \{1, ..., l\}$. The mixing rule employed for the "composite" finger is

$$m_{i,\,j \rightarrow \{1,\,...,\,l\}} = \frac{1}{2}\left(\prod_{p=1}^{l} m_{i,\,j \rightarrow p} + \max_{p \in \{1,\,...,\,l\}} m_{i,\,j \rightarrow p}\right)$$

This rule makes it more difficult for "composite" fingers to be grouped together as virtual fingers. The steps for determining the number of virtual fingers and their constituents proceed as for the previous examples. An example of how the mixing rule is used is shown in Fig. 29.



$\theta = 30°$

$m_{1,\,2 \rightarrow 1} = 0.07$

$m_{1,\,2 \rightarrow 2} = 0.07$

$m_{1,\,2 \rightarrow \{1,\,2\}} = \frac{1}{2}(4.9 \times 10^{-3} + 0.07)$

$= 0.0375$

Fig. 29 **Illustration for mixing rule application**

5.6.3 Hand Configuration Notation

A shorthand notation for the configuration of the hand, say, $VF_1 = \{1\}$, $VF_2 = \{2, 3, 4\}$ is {(1)(234)}. This notation will be used in the remainder of this thesis.

5.6.4 General Mixing Rule for "Composite" Fingers

The virtual finger membership index between two "composite" fingers i and j is given by the expression

$$m_{i \to \{1, \ldots, l_i\}, j \to \{1, \ldots, l_j\}} =$$

$$\frac{1}{2}\left(\prod_{q=1}^{l_j} \max_{p \in \{1, \ldots, l_i\}} m_{i \to p, j \to q} + \prod_{p=1}^{l_i} \max_{q \in \{1, \ldots, l_j\}} m_{i \to p, j \to q} \right)$$

The expression on the right is easily seen to be commutative in "composite" fingers i and j. This mixing rule has a simple physical interpretation: Each contact point on one "composite" finger is only matched to its nearest equivalent (in terms of force vector) and not to all other points in the other "composite" finger. For each contact point of finger i, the largest membership index with finger j is found and then multiplied together; the other term is determined by interchanging these fingers.

We see that this expression reduces to the equation in section 5.6.2 when $l_i = 1$. If the two "composite" fingers are fully compatible, i.e., $l_i = l_j = l$ (say) and $m_{i \to p, j \to q} = 1$ for p, q = 1, ..., l, then $m_{i \to \{1, \ldots, l_i\}, j \to \{1, \ldots, l_j\}} = 1$, as to be expected.

5.7 Experiments and Results

5.7.1 Analysis of Grasps by Human Subjects

Experiments Involving Precision Grasps

Several experiments were conducted to illustrate the use of the mathematical framework for both precision and power grasps. Since force is not measured for these simple experiments, a strong assumption is made here: The force exerted at each finger is the same. This means that the sum of the forces at the contact points of each each finger is equal. So, if there are two contact points at the thumb, then the force at each thumb contact is equal to 1.5 times the force at each contact point on the other fingers. Also, note that only the contact points of the fingers and/or palm with the grasped object are considered in the analysis. Contact points of the finger on the hand itself are not taken into consideration; e.g., thumb contact

with the index finger in the “coal-hammer” grasp are disregarded. In practice, these “invalid” contact points can be determined by analyzing the hand configuration.

Table 3 Description of experiments involving precision grasps

Expt. #	Description
1	Four fingers and thumb on flat rectangular object (3.0 cm x 16.0 cm)
2	Three fingers and thumb on flat rectangular object (3.0 cm x 16.0 cm)
3	Two fingers and thumb on flat small circular object (diameter = 2.4 cm)
4	Four fingers and thumb on flat circular object (diameter = 5.8 cm)
5	Three fingers and thumb on flat circular object (diameter = 5.8 cm)
6	Three fingers and thumb on flat small right triangle (sides 5.2 cm, 6.7 cm , and 8.4 cm)
7	Four fingers and thumb on flat right triangle (sides 10.0 cm, 8.8 cm, and 13.3 cm)
8	Three or four fingers and thumb on flat elliptical object (a = 3.4cm, b = 2.0cm)[a]
9	Three or four fingers and thumb on flat elliptical object (a = 4.0cm, b = 2.0cm)

a. a and b are the semi-major and semi-minor axis lengths respectively.

Table 4 Results of experiments involving precision grasps

Expt. #	$C_{eff,max}$	$Config_{max}$	$C_{eff,2}$	$Config_2$
1	0.707	{(1)(2345)}	0.550	{(1)(2)(345)}
2	0.707	{(1)(234)}	0.550	{(1)(2)(34)}
3	0.550	{(1)(2)(3)}	0.528	{(1)(23)}
4	0.538	{(1)(2345)}	0.514	{(1)(234)(5)}
5	0.578	{(1)(234)}	0.518	{(1)(23)(4)}
6	0.561	{(1)(234)}	0.550	{(1)(23)(4)}
7	0.595	{(1)(2345)}	0.550	{(1)(2)(345)}
8	0.681	{(1)(234)}	0.545	{(1)(23)(4)}
9	0.680	{(1)(2345)}	0.550	{(1)(234)(5)}

In the tables of results, the numbers 0 and 1 refer to the palm and thumb respectively, while the numbers 2, 3, 4, and 5 refer to the other four fingers in order (2 being the index finger and 5 being the little finger). Table 4 shows the results of the experiments described in Table 3 for a subject. $C_{eff,max}$ is the maximum grasp cohesive index while $C_{eff,2}$ is the second largest grasp cohesive index found for the hand configuration. Experiments 3, 4, and 5 (with

flat circular objects) are repeated for five other subjects, and their results are shown in Table 5. As Table 5 indicates, the best cohesive indices do not exhibit a strong consistency in values across different subjects. The optimal hand configuration depends on the manner upon which the object is grasped. For example, in the tripod grasp, if the object is grasped such that the index and middle fingers are separated relatively far apart, as were most of the subjects' hands in the experiments, then they are regarded as separate virtual fingers. If, on the other hand, these fingers are kept close to one another, then they will be grouped as one virtual finger, as for subject 5 (Table 5).

Table 5 Best cohesive indices for precision grasps on flat circular objects

Expt. #	3	4	5
Subject #	$C_{eff,max}$	$C_{eff,max}$	$C_{eff,max}$
0	0.550 {(1)(2)(3)}	0.538 {(1)(2345)}	0.578 {(1)(234)}
1	0.550 {(1)(2)(3)}	0.498 {(1)(2)(345)}	0.497 {(1)(23)(4)}
2	0.550 {(1)(2)(3)}	0.523 {(1)(2345)}	0.526 {(1)(234)}
3	0.550 {(1)(2)(3)}	0.492 {(1)(2)(345)}	0.512 {(1)(23)(4)}
4	0.550 {(1)(2)(3)}	0.483 {(1)(2)(345)}	0.510 {(1)(23)(4)}
5	0.574 {(1)(23)}	0.498 {(1)(2)(345)}	0.514 {(1)(23)(4)}

Table 6 lists the results for Experiments 8 and 9 which involves flat elliptical objects of different eccentricities. In both these experiments, the thumb is considered as one virtual finger and the other fingers as the other virtual finger. While most subjects used five fingers in handling these objects, several of them used only four fingers.

Table 6 Best cohesive indices for precision grasps on flat elliptical objects

Expt. #	8	9
Subject #	$C_{eff,max}$	$C_{eff,max}$
0	0.681 {(1)(234)}	0.680 {(1)(2345)}
1	0.607 {(1)(2345)}	0.655 {(1)(2345)}
2	0.660 {(1)(2345)}	0.680 {(1)(2345)}
3	0.666 {(1)(2345)}	0.684 {(1)(2345)}
4	0.679 {(1)(234)}	0.658 {(1)(2345)}
5	0.664 {(1)(2345)}	0.687 {(1)(2345)}

Table 6 Best cohesive indices for precision grasps on flat elliptical objects

Expt. #	8	9
Subject #	$C_{eff,max}$	$C_{eff,max}$
6	0.620 {(1)(234)}	0.673 {(1)(234)}
7	0.655 {(1)(234)}	0.680 {(1)(2345)}
8	0.663 {(1)(234)}	0.676 {(1)(234)}
9	0.648 {(1)(2345)}	0.676 {(1)(2345)}
10	0.578 {(1)(2345)}	0.670 {(1)(2345)}
11	0.657 {(1)(234)}	0.645 {(1)(2345)}

Experiments Involving Power Grasps

Eight experiments were performed to illustrate the use of the mathematical framework in determining the number and composition of virtual fingers in power grasps. The experiments involved marking the centers of each phalangeal segment and palm, and applying different types of power grasps on rods of different thicknesses and a sphere. Description of the experiments are listed in Table 7. It is assumed, for simplicity of this analysis, that the effective forces at each contact point are normal to the object surface.

The “coal-hammer” grasp is a special case of the cylindrical power grasp, and is identified by the high degree of thumb abduction. We define the type 1 “coal-hammer” grasp to be one in which the thumb does not touch the held object, while the type 2 “coal-hammer” grasp refers to one in which the thumb touches the object. The type 2 grasp normally occurs for a thick object, as in the case of experiment 2 described in Table 7.

Table 7 Description of experiments involving power grasps

Expt. #	Description
1	Spherical power grasp. Radius of sphere = 3.26 cm
2	Type 2 cylindrical “coal-hammer” grasp (thick cylinder). Radius of circular cross-section = 3.30 cm.
3	Type 1 cylindrical “coal-hammer” grasp (medium-thick cylinder). Radius of circular cross-section = 1.47 cm. (Note: the thumb does not touch the cylinder)
4	Cylindrical power grasp (medium-thick cylinder). Radius of circular cross-section = 1.47 cm.
5	Type 1 cylindrical “coal-hammer” grasp (thin cylinder). Radius of circular cross-section = 0.97 cm. (Note: the thumb does not touch the cylinder).

Table 7 Description of experiments involving power grasps

Expt. #	Description
6	Cylindrical power grasp (thin cylinder). Radius of circular cross-section = 0.97 cm.
7	Cylindrical power grasp (elliptical cross-section with a = 3.3cm, b = 1.9cm)[a]
8	Cylindrical power grasp (elliptical cross-section with a = 4.1cm, b = 1.9cm)

a. a and b are the semi-major and semi-minor axis lengths respectively.

Table 8 Results of experiments involving precision grasps

Expt. #	$C_{eff,max}$	$Config_{max}$	$C_{eff,2}$	$Config_2$
1	0.407	{(0)(1)(2345)}	0.318	{(0)(1234)(5)}
2	0.542	{(0)(1)(2345)}	0.270	{(0)(12345)}
3	0.666	{(0)(2345)}	0.544	{(02345)}
4	0.531	{(0)(1)(2345)}	0.337	{(0)(12345)}
5	0.650	{(0)(2345)}	0.528	{(0)(34)(25)}
6	0.522	{(0)(1)(2345)}	0.351	{(0)(12345)}
7	0.538	{(0)(1)(2345)}	0.427	{(0)(12345)}
8	0.537	{(0)(1)(2345)}	0.415	{(0)(12345)}

It is interesting to note from Table 8 that, despite the differences in cylinder thickness, the maximum grasp indices for the power grasps in experiments 2, 4 and 6 do not differ very much from one another. It is also interesting to note that the grasp cohesive index remains about the same despite changes in the cross-sectional shapes, as evidenced in the results of Experiments 7 and 8. The corresponding values for the “coal-hammer” grasps for experiments 3 and 5 do not seem to be significantly different from each other.

Table 9 Best cohesive indices for power grasps (including Type 2 “coal-hammer” grasps) on cylinders of different thicknesses

Subject #	$C_{eff,max}$ (thick cylinder)	$C_{eff,max}$ (medium cylinder)	$C_{eff,max}$ (thin cylinder)
0	0.542	0.531	0.522
1	0.534	0.531	0.526
2	0.543	0.534	0.505
3	0.541	0.533	0.535

Table 9 Best cohesive indices for power grasps (including Type 2 “coal-hammer” grasps) on cylinders of different thicknesses

Subject #	$C_{eff,max}$ (thick cylinder)	$C_{eff,max}$ (medium cylinder)	$C_{eff,max}$ (thin cylinder)
4	0.536	0.531	0.529
5	0.543	0.538	0.528
6	0.542	0.524	0.522
7	0.544	0.535	0.453
8	0.543	0.534	0.532
9	0.544	0.511	0.515
10	0.537	0.535	0.513
11	0.545	0.516	0.497

Table 9 shows the best grasp cohesive index for the power grasps on cylinders of different thickness. Note that the type 2 “coal-hammer” grasp is virtually unidentifiable as a special case of the power grasp on the basis of the grasp cohesive index alone. This is due to the thumb touching the object. The virtual configuration and composition for all these grasps and for all the different subjects is the same, namely, {(0)(1)(2345)}. The average value is 0.528, with the standard deviation of 0.017 (3.2% of the average value).

Table 10 Best cohesive indices for Type 1 “coal-hammer” grasps on cylinders of different thicknesses

Subject #	$C_{eff,max}$ (medium cylinder)	$C_{eff,max}$ (thin cylinder)
0	0.666	0.650
1	0.681	0.665
2	0.676	0.661
3	0.677	0.640
4	0.672	0.672
5	0.680	0.645
6	0.681	0.661
7	0.678	0.657
8	0.659	0.683
9	0.671	0.683

Table 10 Best cohesive indices for Type 1 "coal-hammer" grasps on cylinders of different thicknesses

Subject #	$C_{eff,max}$ (medium cylinder)	$C_{eff,max}$ (thin cylinder)
10	0.683	0.676
11	0.646	0.627

Table 10 lists the results for the type 1 "coal-hammer" grasps on two cylinders of differing thicknesses. Again the best effective cohesive index is relatively independent of the thickness of the cylinder grasped and the person holding the object. The mean best grasp cohesive index is 0.666 with the standard deviation of 0.016 (2.4% of the average index). The configuration associated with all the indices is {(0)(2345)}.

Table 11 Best cohesive indices for power grasps on cylinders with elliptical cross-section of different eccentricities

Subject #	$C_{eff,max}$ (a=3.3cm; b=1.9cm)	$C_{eff,max}$ (a=4.1cm; b=1.9cm)
0	0.538	0.537
1	0.538	0.538
2	0.537	0.540
3	0.535	0.543
4	0.535	0.535
5	0.536	0.536
6	0.539	0.540
7	0.537	0.538
8	0.534	0.524
9	0.538	0.538
10	0.539	0.534
11	0.537	0.528

The virtual finger configuration and composition for all the subjects is {(0)(1)(2345)}. The mean grasp cohesive index is 0.536 and the standard deviation is 3.8×10^{-3}, which is about 0.7% of the mean. (Note: *a* and *b* in Table 11 are the semi-major and semi-minor axis lengths respectively.) The mean grasp cohesive index for these grasp experiments is virtually the same as that for grasp experiments involving cylinders of circular cross-sections. This further strengthens our claim that the grasp cohesive index can be used to identify prismatic power grasps, regardless of the cross-sectional shape of the object held. The similarity

in the grasp cohesive indices can be attributed to the proximity of the four fingers to each other and the high similarity in the finger configuration in this type of power grasp.

Table 12 Best effective cohesive indices for spherical power grasps

Subject #	$C_{eff,max}$	$C_{eff,2}$
0	0.407 {(0)(1)(2345)}	0.318 {(0)(1234)(5)}
1	0.400 {(0)(1)(234)}	0.329 {(0)(123)(4)}
2	0.375 {(0)(1)(2345)}	0.294 {(0)(1234)(5)}
3	0.343 {(0)(123)(45)}	0.329 {(0)(1)(2345)}
4	0.361 {(0)(1234)(5)}	0.310 {(0)(1)(2345)}
5	0.447 {(0)(1)(2345)}	0.364 {(0)(1234)(5)}
6	0.326 {(0)(123)(45)}	0.311{(0)(1234)(5)}
7	0.444 {(0)(1)(2345)}	0.353 {(0)(1234)(5)}
8	0.493 {(0)(1)(2345)}	0.449 {(0)(12345)}
9	0.488 {(0)(1)(2345)}	0.349 {(0)(12345)}
10	0.393 {(0)(1)(2345)}	0.366 {(0)(1234)(5)}
11	0.426 {(0)(1)(2345)}	0.312 {(0)(1234)(5)}

Similar experiments were conducted using a sphere. The results for the spherical power grasps are shown in Table 12. The mean best grasp cohesive index is 0.409 with the standard deviation of 0.051 (12.5% of the mean). As can be seen, the best grasp cohesive indices differ markedly from person to person, and even then, the configuration associated with the best cohesive index is different for subjects 1, 3, 4, and 6. For the others, the configuration is {(0)(1)(2345)}. Note that for subject 1, the last or little finger barely touched the ball and hence was not considered in the analysis, and the configuration is {(0)(1)(234)} instead. These results are based on the assumption that the force exerted by each finger is the same. If the force data were available and that the thumb exerted a higher force than other fingers (a possible scenario), then the optimal configuration {(0)(1)(2345)} may have been more consistent for all subjects. This prediction has yet to be tested.

The reason for the higher disparity in the grasp cohesive index in spherical grasps than in cylindrical grasps is that in cylindrical grasps, the fingers (excluding the thumb) are normally kept very close together. The relative inter-phalange arrangements in the cylindrical grasps are consistent, despite the differing sizes of the cylinders handled. The amount of finger flexion (due to the different cylinder sizes) has little effect on the grasp cohesive index. For spherical grasps, however, relative inter-phalange arrangements do differ markedly

from subject to subject, causing the higher range of grasp cohesive indices observed. It is interesting to note that the grasp cohesive indices for the spherical grasps are all lower than those for the cylindrical grasps. This is to be expected, because in a spherical grasp, the amount of force interaction between fingers is higher than that in a prismatic power grasp. The higher force interaction is, in turn, attributed to the more widely separated fingers in a spherical grasp and the significant curvature of the sphere.

5.7.2 Procedure for Grasp Recognition

From this study, a grasp can be identified from the following general steps:

1. *Compute the real finger to virtual finger mapping which yields the virtual finger compositions and the grasp cohesive index.*
2. *If the palm surface is not involved in the grasp, classify it as a non-volar grasp.*
3. *Otherwise, by checking the grasp cohesive index and, if necessary, the degree of thumb abduction, classify it either as a spherical, cylindrical or coal-hammer (type 1 or type 2) power grasp.*

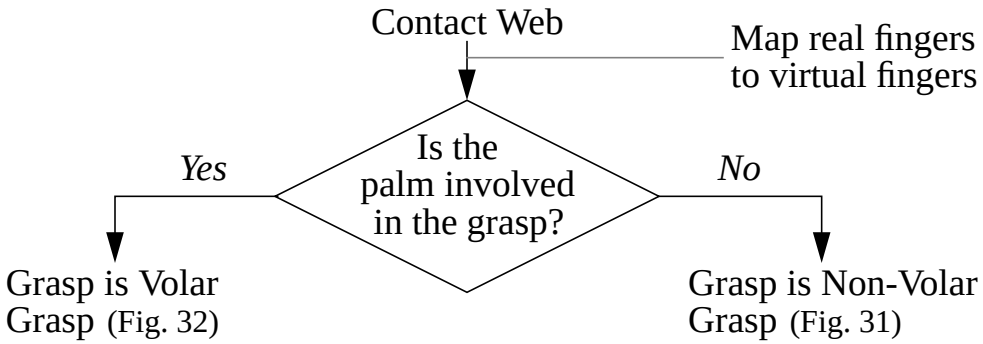


Fig. 30 **Recognition of major type of grasp**

A grasp can be classified as a volar grasp or non-volar grasp according to whether there is volar-object interaction or not (Fig. 30). If it is a non-volar grasp, further classification can be done by checking if only the fingertips are involved in the grasp, and the contact points' closeness of fit to a circle or rectangle. This is illustrated in Fig. 31. Unless the grasp is a lateral pinch (in which case the grasp is a power grasp), the grasp is classified as a precision grasp.

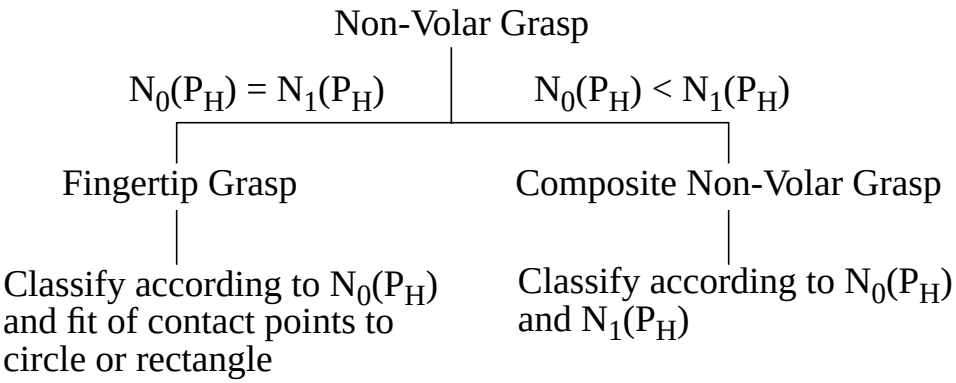


Fig. 31 **Discrimination graph for non-volar grasps**

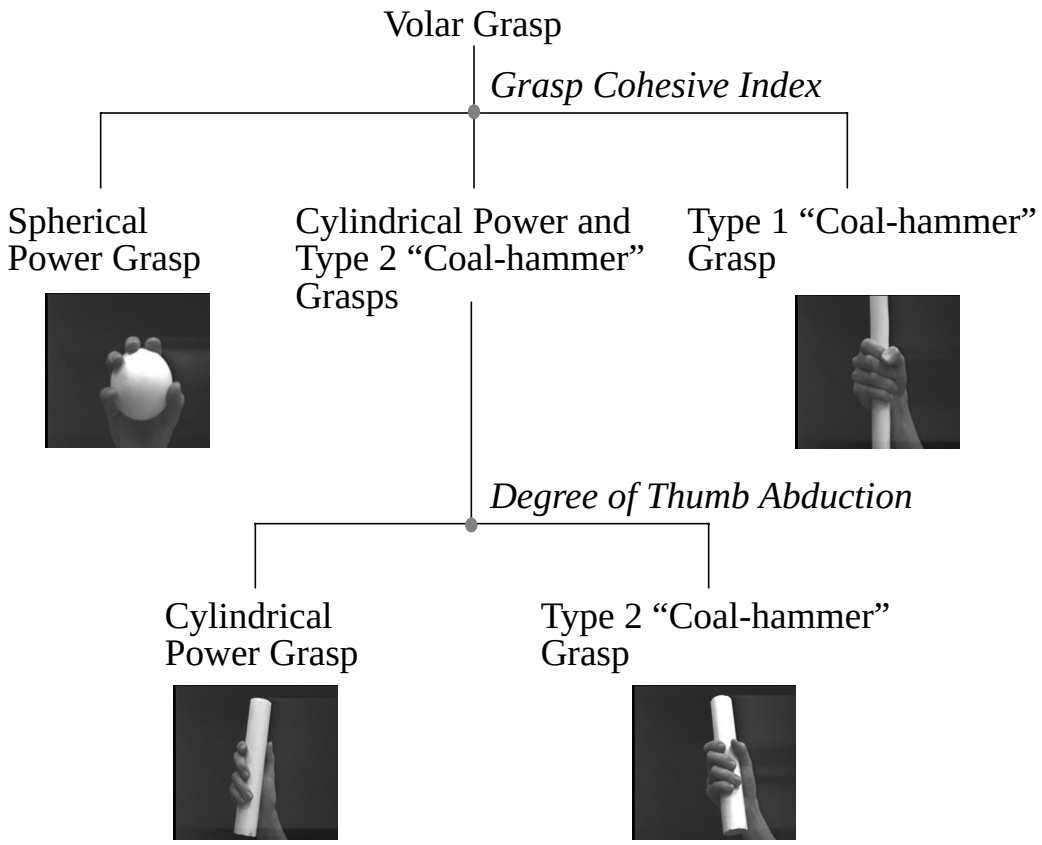


Fig. 32 **Discrimination graph for power grasps**

The volar grasp discrimination procedure in step 3 is graphically depicted in Fig. 32. (Note that all volar grasps are power grasps.) The first level of classification is performed using the following discrimination function:

$$\tau_i = e^{-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma_i}\right)^2}$$

where μ_i is the mean value of the grasp cohesive index for the *i*th power grasp category and σ_i is the associated standard deviation. The power grasp category is identified by the largest value of the discrimination function. Should the cylindrical and type 2 “coal-hammer” grasps need to be discriminated, we would then determine the degree of thumb abduction. The type 2 “coal-hammer” grasp is associated with a high degree of thumb abduction. We use the thumb in a standard position (fully extended and in the plane of the palm) as a reference line in determining the degree of thumb abduction. We consider deviations greater than 45^o to be significant enough to be categorized as a type 2 “coal-hammer” grasp.

Note that the object shape has not been directly taken into consideration here; the local object shape (i.e., part of the object within the compass of the hand) has been implicitly taken care of by the contact web.

5.7.3 Grasp Recognition from Range and Intensity Images

Three range and intensity image sequences of finger movements leading to different power grasps were taken and then analyzed using the grasp recognition scheme described earlier. In each sequence, the fingers are tracked to the final grasp configuration before the grasp is identified. However, prior to this, the hand model needs to be initialized. This is done using a separate program.

Hand Model Initialization

Each finger segment is modeled by a cylinder. The purpose of the hand model initialization is to determine the finger segment cross-sectional radius and length, and the relative positions of the fingers. The assumed hand posture is with the fingers fully extended, and such that the plane of the palm is approximately perpendicular to the camera viewing direction. These fingers are taken to be the reference positions, and abduction angles are measured relative to these positions.

In addition, to facilitate the measurement of finger segment lengths, dark lines were drawn across the finger at the distal and proximal interphalangeal joints (except for the thumb, where a line was drawn across it at the interphalangeal joint). The proximal finger segment length is calculated from empirical anthropometric studies of human finger segment length ratios [3].

The steps involved in hand model initialization are:

1. *Thresholding and hand boundary extraction*

   The image is first thresholded by assigning background values to intensity pixels whose corresponding range values are inadmissible and assigning foreground values if they are admissible. Small regions are eliminated, leaving the hand in the resulting binary image. Subsequently, the hand boundary is extracted using a simple 8-connected boundary following algorithm.

2. *Curvature analysis of hand boundary*

   Using the convention that a convex portion of a body has negative curvature, the tips of the fingers are located at points where curvature minima are observed. Similarly, the five grooves between fingers are located at positions of curvature maxima.

3. *Identification of finger regions*

   The finger regions are identified by noting that if the fingers are cyclically ordered anticlockwise, the thumb is farthest away from the fingers immediately preceding and following it.

4. *Location of interphalangeal joints and calculation of finger segment lengths*

   The position of the interphalangeal joints are approximated by using the Hough transform to locate dark lines. The finger segment lengths are then calculated from the distances between joints or between fingertips and the nearest joints.

5. *Determination of best fit cylinders*

   3-D cylinders are fitted to the fingers from the hand range data using an iterative least-square formulation.

Fig. 33 shows the intensity image of a hand and three snapshots of the hand model initialization program. Note that this program is run only once for a subject's hand.

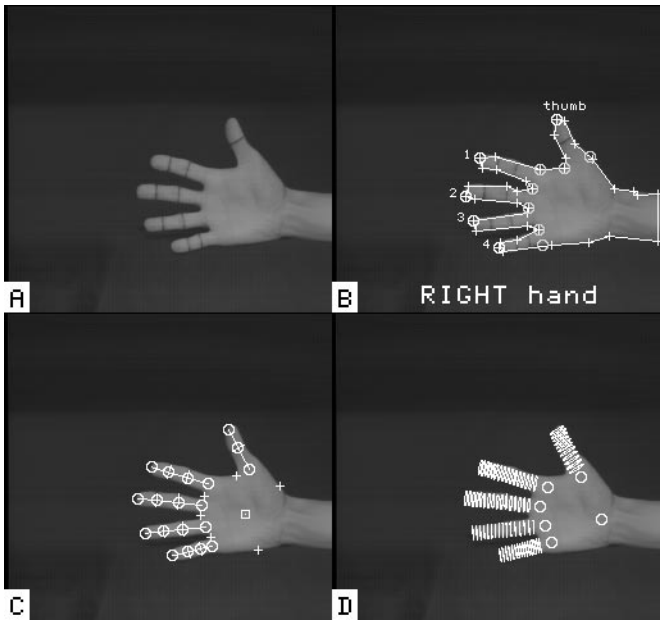


Fig. 33 **Hand model initialization. (a) Intensity image; (b) Identification of fingers; (c) Localization of finger joints; (d) Cylindrical fitting of fingers.**

Finger Tracking

The basic method used in finger tracking is local search of the minimum sum of two types of matching errors: error in range data fitting, and error in matching the hypothesized finger 2-D projection to the actual finger position in the image. The following assumptions made are:

1. *The hand does not move; only the fingers move (via flexion, extension, abduction and adduction).*
2. *The first frame features a hand with fingers fully or nearly fully extended.*
3. *There is no significant interphalangeal occlusion.*

Grasp Recognition Results

Three range and intensity images were recorded and analyzed.

Example 1 (Fig. 34 and Fig. 35)

Four of the eight frames for this grasp sequence is shown in Fig. 34.

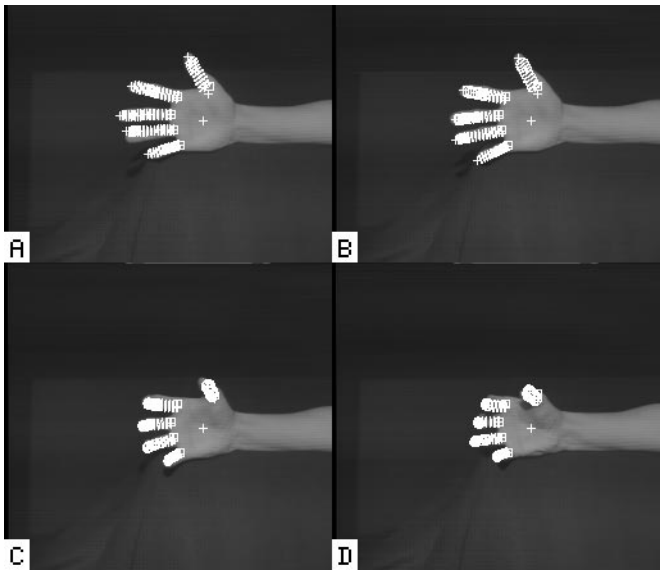


Fig. 34 **Finger tracking sequence for Example 1. (a) Frame 1; (b) Frame 3; (c) Frame 6; (d) Frame 8**

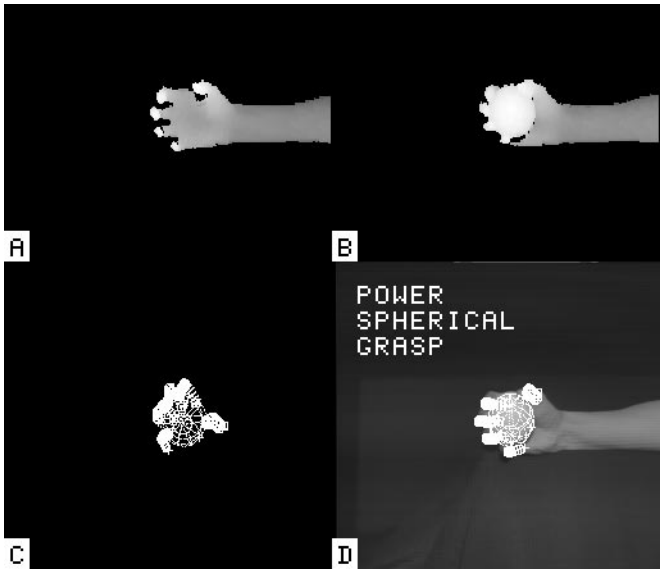


Fig. 35 **Recognition results for a spherical power grasp. (a) Range image of last frame of sequence; (b) Range image of hand and object; (c) Alternate view of tracked fingers with object; (d) Classification of grasp**

The grasp cohesive index for this example is 0.356, with the following virtual finger compositions: $VF_1 = \{0\}$, $VF_2 = \{1\}$, and $VF_3 = \{2, 3, 4, 5\}$. This grasp is classified as a spherical power grasp (Fig. 35).

Example 2 (Fig. 36 and Fig. 37)

Part of the frame sequence for this example is shown in Fig. 36.

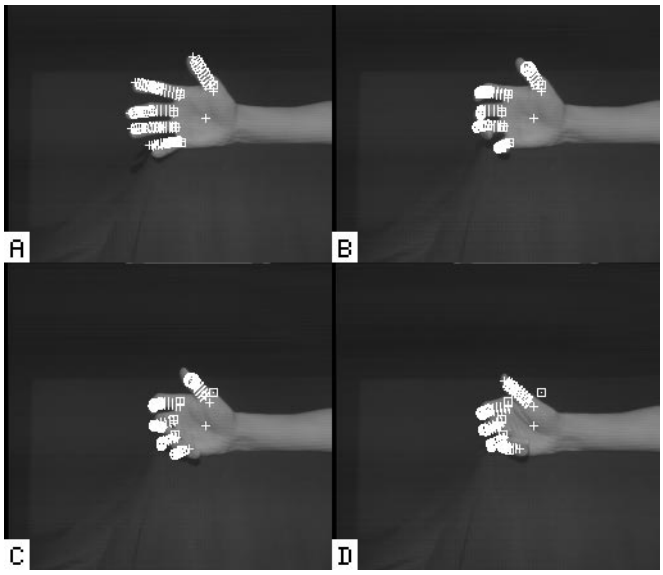


Fig. 36 **Finger tracking sequence for Example 2. (a) Frame 1; (b) Frame 3; (c) Frame 6; (d) Frame 8**

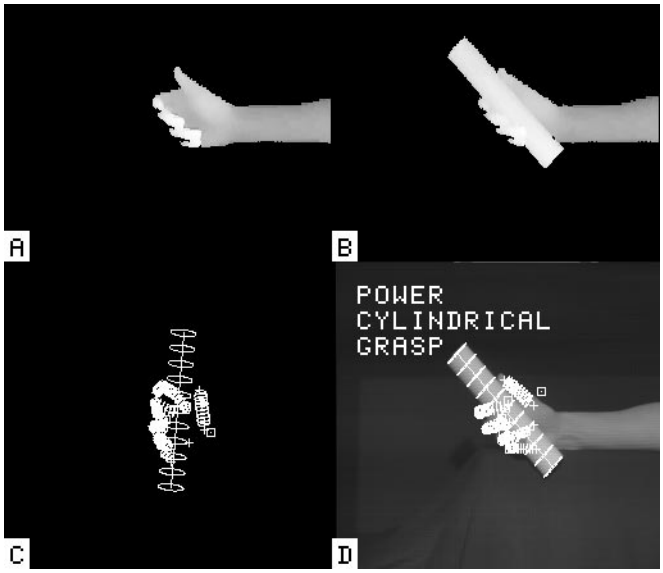


Fig. 37 **Recognition results for a cylindrical power grasp. (a) Range image of last frame of sequence; (b) Range image of hand and object; (c) Alternate view of tracked fingers with object; (d) Classification of grasp**

The grasp cohesive index for the second example is 0.508, with the following virtual finger compositions: $VF_1 = \{0\}$, $VF_2 = \{1\}$, and $VF_3 = \{2, 3, 4, 5\}$. From the grasp cohesive index, this grasp can either be a cylindrical power or type 2 “coal-hammer” grasp. Since the angle of the thumb subtends only 23^o with the standard (original) thumb posture, it is classified as a cylindrical power grasp.

Example 3 (Fig. 38 and Fig. 39)

Fig. 38 shows four frames of the grasp sequence for Example 3.

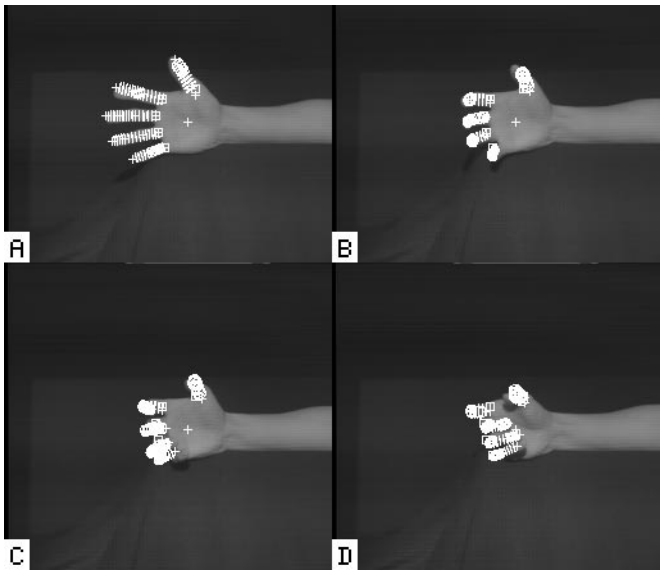


Fig. 38 **Finger tracking sequence for Example 3. (a) Frame 1; (b) Frame 3; (c) Frame 6; (d) Frame 8**

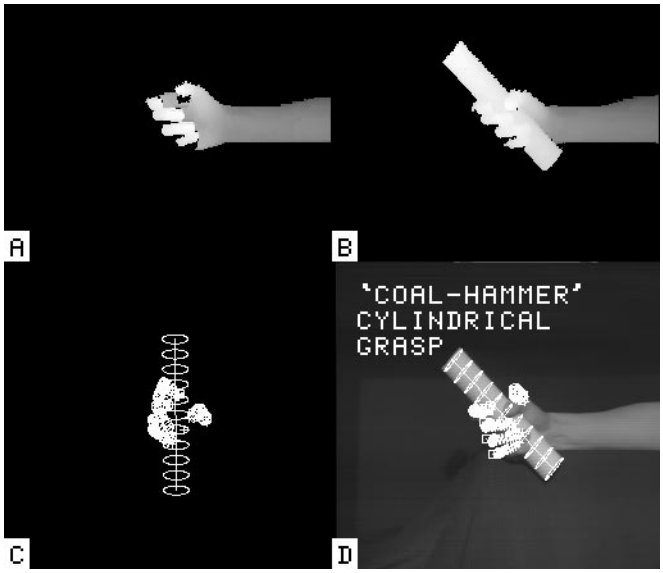


Fig. 39 **Recognition results for a type 2 "coal-hammer" grasp. (a) Range image of last frame of sequence; (b) Range image of hand and object; (c) Alternate view of tracked fingers with object; (d) Classification of grasp**

The grasp cohesive index for this example is 0.527, with the following virtual finger compositions: $VF_1 = \{0\}$, $VF_2 = \{1\}$, and $VF_3 = \{2, 3, 4, 5\}$. As with example 2, just from the grasp cohesive index alone, this grasp could be either a cylindrical power or a type 2 "coal-hammer" power grasp. From the high degree of thumb abduction (subtending 77^o with the original thumb configuration), it is thus classified as a type 2 "coal-hammer" grasp.

The experiments and their results described in sections 5.7.1 and 5.7.3 indicate that it is possible to categorize grasps by using the contact web and real finger-to-virtual finger mapping. This mapping is instrumental in characterizing the type of grasp demonstrated in the scene.

5.8 Abstraction Hierarchy of a Grasp

With the preceding definitions, we now describe the proposed grasp abstraction hierarchy shown in Fig. 40. In this hierarchy, the grasp is described using a hierarchy of hand elements of increasing conceptual complexity. Each element is an entity containing, at its appropriate level, a vector of internal states. The hand elements can be finger segments and joints, physical fingers and palm, virtual fingers, or opposition spaces. The finger segments and joints are the lowest-level elements which describe the low-level hand configuration. The finger segment description includes its approximate 3-D shape and pose, its contact point with the object (if there is contact) and the force vector at that contact point. The joint element contains the degree of flexion as well as the abduction angle, if applicable. Each element is "active" if all or part of it is directly in contact with the object, and "inactive" otherwise. The elements described here are implemented using frames.

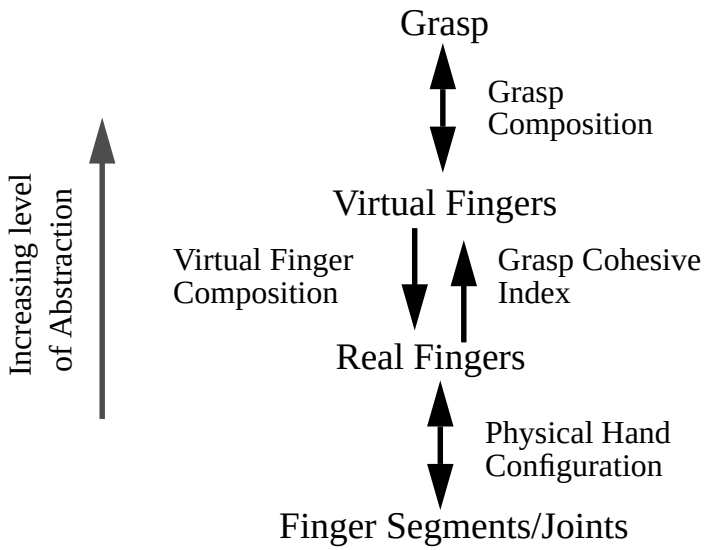


Fig. 40 **Abstraction hierarchy of a grasp**

This hierarchical scheme is not unlike that described in [56]; the virtual fingers correspond to the "opposition space level" while the real finger and segment agents correspond to the "biomechanical level." The proposed abstraction hierarchy will be used in the task-sensitive matching of real human fingers to manipulator fingers.

The abstraction hierarchy is influenced in part by the analogy to the three proposed categories of human behavior: skill-based, rule-based, and knowledge-based performance [127]. The types of information associated with these behavior are signals, signs, and symbols,

respectively. Skill-based behavior is at the lowest level, with acts taking place without conscious control as smooth, automated and highly integrated patterns of behavior. Rule-based behavior comprises a sequence of actions controlled by stored rules; knowledge-based behavior constitutes action based on high-level knowledge and goals. Signals, signs, and symbols are the names given to the form of information used and interpreted at these respective levels. For the grasp abstraction hierarchy, the segment and real finger elements correspond to the signals, the virtual finger and opposition space elements the signs, and the grasp the symbols (Fig. 40). Humans relate most comfortably with symbols; signals in the form of raw data are least likely to be useful to the human, especially if the human is not an expert. The abstraction hierarchy provides a mechanism for the operator to judge the efficacy of the recognition process and act accordingly (if intervention is deemed necessary) based on this hierarchy.

5.8.1 Examples of Grasp Abstraction Hierarchies

We illustrate the hierarchical description of grasps with the following three examples: a cylindrical power grasp[1], a type 2 coal-hammer grasp[2], and a precision tripod grasp. (In Fig. 41, Fig. 43, and Fig. 45, A_0 represent the palm while A_{ij} represent the *j*th segment of the *i*th finger. In Fig. 42, Fig. 44, and Fig. 46, the *i*th virtual finger is indicated by VF_i and the *i*th real finger by F_i.)

Example 1: Cylindrical Power Grasp

A cylindrical power grasp is characterized by flexion in unison of the four fingers and a low degree of thumb abduction (Fig. 41). The hierarchical description of this grasp is shown in Fig. 42. The virtual finger compositions are determined by employing the real finger-to-virtual finger mapping described earlier. The primary type of opposition in this grasp is palm opposition between VF0 (palm) and VF2 (four fingers). While there is little force interaction between the thumb and the palm, there is some between the thumb and the four fingers. Other information such as the internal state vector of each segment and real finger agents are not shown here.

---

1. In this grasp, the object is held within the compass of the hand with all the fingers and palm being in contact with the object. The thumb is adducted, i.e., placed on the cylinder in such a manner that the thumb is almost parallel to the axis of the cylindrical object held.
2. A coal-hammer grasp is similar to the cylindrical power grasp, except that the thumb is significantly abducted, i.e., placed at an angle to the cylindrical object axis. This lends more power to the grasp at the expense of dexterity. A Type 1 coal-hammer grasp arises when the thumb does not make contact directly with the object; it rests on the dorsal aspects of the index and/or middle finger. In a Type 2 coal-hammer grasp, however, the thumb touches the held object. Whether a coal-hammer grasp is Type 1 or Type 2 depends on the relative spans of the hand and the object.

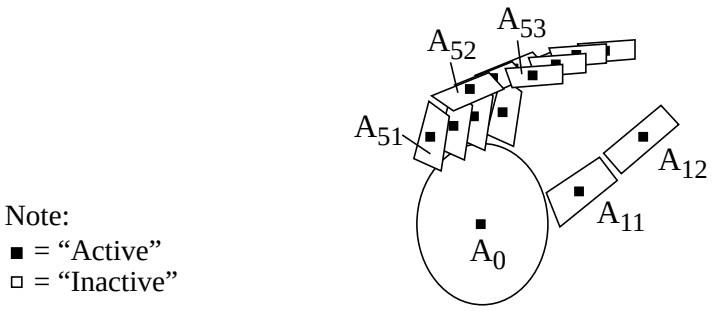


Fig. 41 **Cylindrical power grasp**

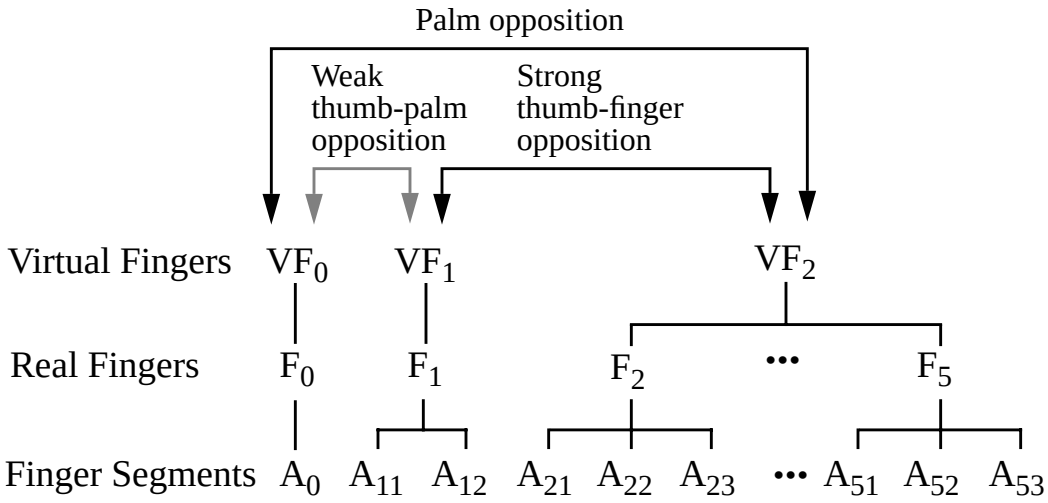


Fig. 42 **Hierarchical decomposition of the cylindrical power grasp**

Example 2: Type 2 “Coal-hammer” Grasp

The difference between the normal cylindrical power grasp and the type 2 “coal-hammer” grasp is the high degree of thumb abduction in the latter. The grasp and its decomposition are depicted in Fig. 43 and Fig. 44 respectively. Due to the posture of the thumb which reinforces the grip on the object at the expense of dexterity, there exists a strong interaction between the thumb and the palm, as well as between the thumb and the four fingers.

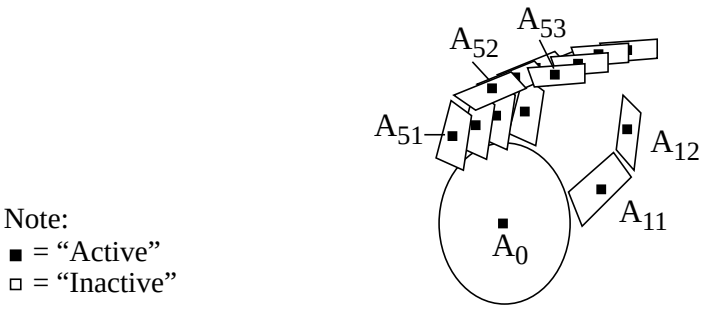


Fig. 43 **Type 2 “coal-hammer” cylindrical power grasp**

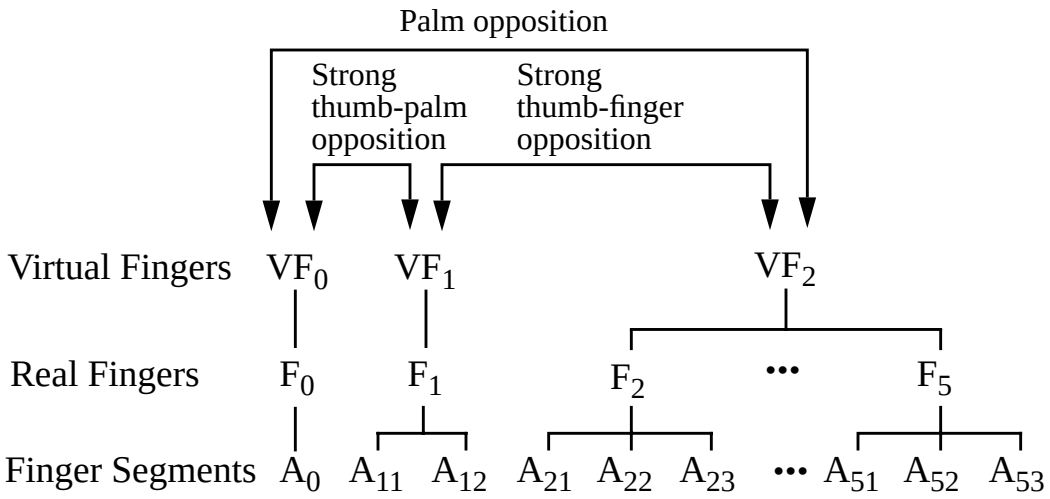


Fig. 44 **Hierarchical decomposition of the Type 2 "coal-hammer" cylindrical power grasp**

Example 3: Tripod Precision Grasp

In this precision grasp, only the tips of the thumb, index finger and middle finger are in contact with the object. The grasp and its decomposition are shown in Fig. 45 and Fig. 46 respectively. Note that in this example, the index and long fingers are assumed to be spaced far apart enough so as to be classified as different virtual fingers.

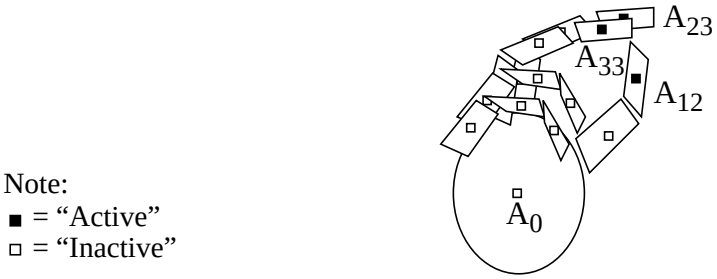


Fig. 45 **Precision tripod grasp**

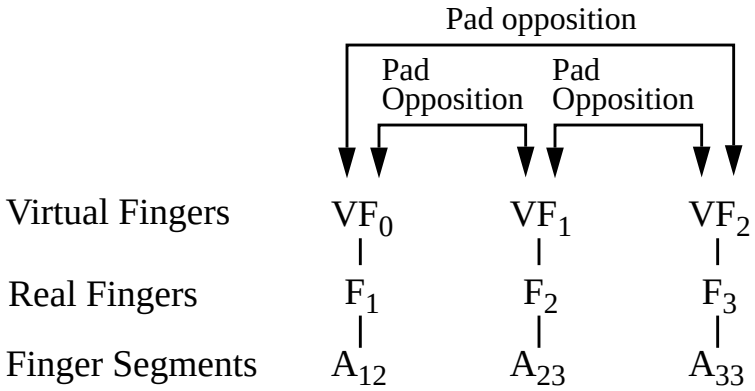


Fig. 46 **Hierarchical decomposition of the precision tripod grasp**

5.8.2 Implementation

We track the configuration (joint angles) and pose (position and orientation) of the hand using the *Cyber*Glove [33] and Polhemus [1] devices respectively. The software which reads in and interprets the hand configuration and pose are mostly written in C (some of which are adapted from the VirtualHand v1.0 software supplied by Virtual Technologies), while that which provides the object representation and its relationship with the hand is written in Common Lisp. The geometric modeler used in our work is Vantage [8]. In addition, the frames created to represent the grasp hierarchy are done using Knowledge Craft [76]. Knowledge Craft is a toolkit for knowledge engineers and AI system builders, and it uses a frame-based knowledge representation language called CRL (Carnegie Representation Languange) with procedural attachment and inheritance.

5.8.3 Results of Experiments

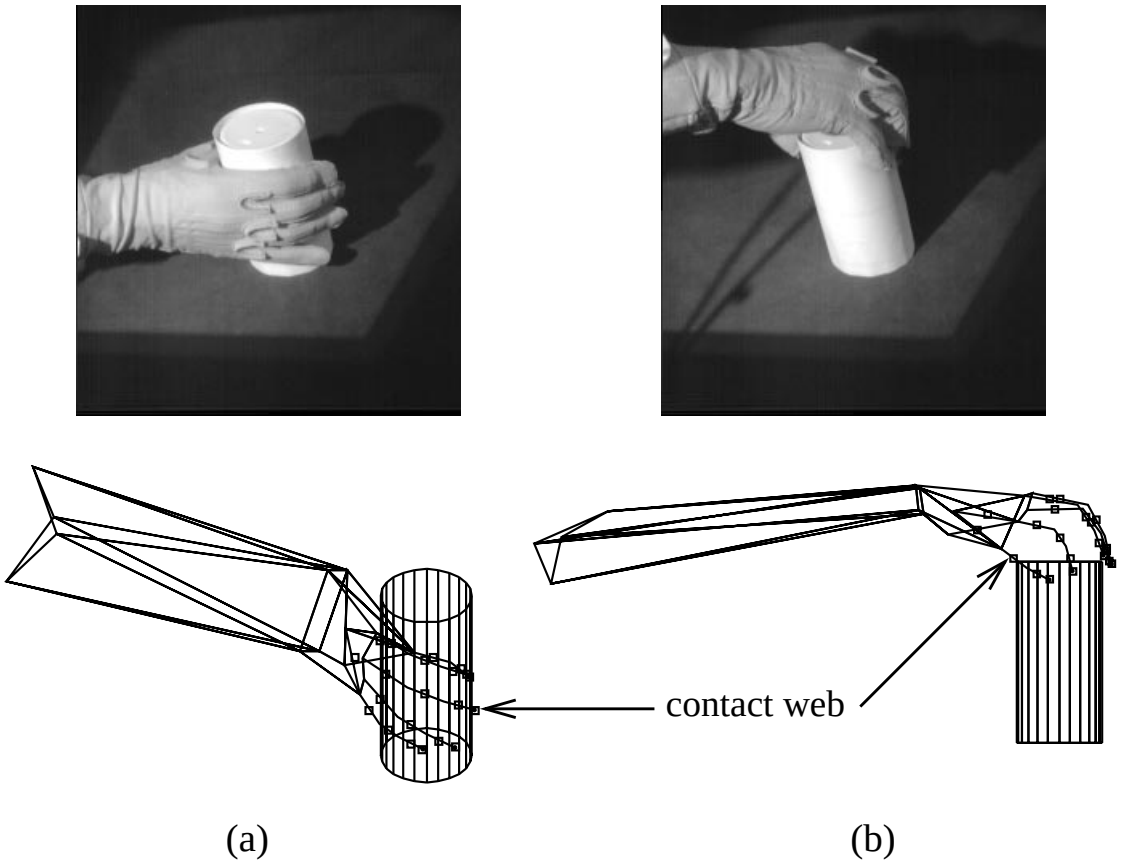


Fig. 47 **Two examples of grasping: (a) a power grasp, (b) a precision grasp. The small boxes indicate the positions of the middle of finger segments, fingertips, and the center of the palm.**

Two experiments, one involving a power grasp and another involving a precision grasp, have been conducted to further illustrate the grasp abstraction hierarchy. The final poses of the grasps are depicted in Fig. 47. In each case, the small squares which represent the contact points of the hand compose the contact web. The poses and configurations of the hand

were obtained using the *Cyber*Glove and Polhemus devices. In both cases, the cylinder was created using the geometric modeler Vantage. The measurements of the cylinder is based on an actual cylindrical object. The positional and orientation data of the object were extracted to determine the relative distances of the hand contact points to the object, as well as the contact normals.

In the first experiment which involves a power grasp (Fig. 47(a)), the grasp recognition program has correctly identified it as a Type 2 ‘Coal-hammer’ cylindrical grasp. Note that the identification is based on the contact normal and configuration of the hand only. The output of this program is as follows:

```
---------------------------------------------------------
Effective cohesive index = 0.535347
Virtual finger #1 = VF0 = { Palm }; Cohesive index = 1.0
Virtual finger #2 = VF1 = { 0 }; Cohesive index = 1.0
Virtual finger #3 = VF2 = { 1 2 3 4 }; Cohesive index = 0.921
---------------------------------------------------------
Contact code is 111111
Grasp is a power grasp.
Gauss values = 0.910846, 0.000000, 0.046480 (see text for explanation)
Thumb abduction angle = 32.0 degs
Grasp is TYPE 2 ‘COAL-HAMMER’ CYLINDRICAL GRASP.
Number of virtual fingers = 3
Number of opposition spaces = 3
Virtual fingers involved in opposition spaces:
opp0: VF0, VF1; Opposition strength = 0.700
opp1: VF0, VF2; Opposition strength = 0.826
opp2: VF1, VF2; Opposition strength = 0.980
```

The effective cohesive index and virtual finger composition were determined using the procedure described earlier. The contact code shows that all the fingers and the palm are active, i.e., in direct contact with the object of interest. The “Gauss values” given above are the extent of similarity to the power cylindrical grasp/Type 2 coal-hammer grasp[1], the Type 1 coal-hammer grasp[2], and the spherical grasp, respectively.

The schemata representing the power grasp are shown below. Note that the joint angles are expressed in degrees while the positional information is in cm. The opposition space schema is denoted as OPP.

---

1. The power cylindrical grasp is differentiated from the Type 2 coal-hammer grasp by the degree of thumb abduction. The former is characterized by negligible thumb abduction (thumb is adducted) while the latter features significant thumb abduction (> 20 degrees).
2. In the Type 1 coal-hammer grasp, the thumb is placed over the dorsal side of the index and/or middle fingers, and does not directly make contact with the object.

{{ **WRIST+HAND**
HAS-MEMBER: WRIST HAND
N-FRAMES: 1
ACTIVE: T}}

{{ **WRIST**
MEMBER-OF: WRIST+HAND
PITCH: -34.2 ; in degrees
YAW: 16.48}}

{{ **HAND**
MEMBER-OF: WRIST+HAND
HAS-MEMBER: FINGER4 FINGER3 FINGER2 FINGER1 FINGER0 OPP2 OPP1 OPP0 VF2 VF1 VF0 VOLAR FINGER
GRASP: "TYPE 2 'COAL-HAMMER' CYLINDRICAL GRASP"
EFFECTIVE-COHESIVE-INDEX: 0.535
ACTIVE: T}}

{{ **VOLAR**
MEMBER-OF: HAND
CONTACT-LOCATION: (5.80 -1.51 -6.37) ; in cm.
HALF-THICKNESS: 1.3
ACTIVE: T}}

The virtual finger and opposition schemata are listed below. The slot *SPAN* of the virtual finger schema indicates the number of fingers (including the palm) in the virtual finger. The value in the *STRENGTH* slot of the opposition schema indicates the degree to which the two virtual fingers opposes each other. The opposition strength of unity would indicate that the two virtual fingers are exerting forces directly against each other. Section 5.9 shows how the opposition strength is calculated. The opposition schemata listed below indicate that the opposition between the virtual finger comprising the thumb and the virtual finger comprising the other four fingers are the most significant. This is reasonable as the cylinder held has a large cross-sectional diameter of about 7 cm.

{{ **VF0**
MEMBER-OF: OPP1 OPP0 HAND
HAS-MEMBER: VOLAR
COHESIVE-INDEX: 1.0
SPAN: 1}}

{{ **VF1**
MEMBER-OF: OPP2 OPP0 HAND
HAS-MEMBER: FINGER0
COHESIVE-INDEX: 1.0
SPAN: 1}}

{{ **VF2**
MEMBER-OF: OPP2 OPP1 HAND
HAS-MEMBER: FINGER4 FINGER3 FINGER2 FINGER1
COHESIVE-INDEX: 0.921
SPAN: 4}}

{{ **OPP0**
MEMBER-OF: HAND
HAS-MEMBER: VF1 VF0
TYPE: "Palm opposition"
STRENGTH: 0.700}}

{{ **OPP1**
MEMBER-OF: HAND
HAS-MEMBER: VF2 VF0
TYPE: "Palm opposition"
STRENGTH: 0.826}}

{{ **OPP2**
MEMBER-OF: HAND
HAS-MEMBER: VF2 VF1
TYPE: "Finger opposition"
STRENGTH: 0.980}}

We list only the finger, joint, and segment schemata for the index finger (**FINGER1**).

Schema for finger 1:

{{ **FINGER1**
INSTANCE: FINGER
MEMBER-OF: VF2 HAND
HAS-MEMBER: FINGER1-J3 FINGER1-J2 FINGER1-J1 FINGER1-J0
FINGER1-S3 FINGER1-S2 FINGER1-S1 FINGER1-S0
RADIUS: 0.81 ; determined from a hand model initialization program described earlier
ABDUCT-ANGLE: 0.3
ACTIVE: T}}

Joint schemata for finger 1:

{{ **FINGER1-J0**
MEMBER-OF: FINGER1
FLEX-ANGLE: 22.3
LOCATION: (3.17 -1.92 -10.11)}}

{{ **FINGER1-J1**
MEMBER-OF: FINGER1
FLEX-ANGLE: 42.6
LOCATION: (-0.91 -2.73 -10.73)}}

{{ **FINGER1-J2**
MEMBER-OF: FINGER1
FLEX-ANGLE: 11.0
LOCATION: (-3.10 -1.66 -10.20)}}

{{ **FINGER1-J3** ; the fingertip
MEMBER-OF: FINGER1
LOCATION: (-4.66 -0.50 -9.60)}}

Segment schemata for finger 1:

{{ **FINGER1-S0**
MEMBER-OF: FINGER1
CONTACT-LOCATION: (1.13 -2.32 -10.42)
ACTIVE: T}}

{{ **FINGER1-S1**
MEMBER-OF: FINGER1
CONTACT-LOCATION: (-2.00 -2.19 -10.46)
ACTIVE: T}}

{{ **FINGER1-S2**
MEMBER-OF: FINGER1
CONTACT-LOCATION: (-3.88 -1.08 -9.90)
ACTIVE: T}}

{{ **FINGER1-S3** ; the fingertip
MEMBER-OF: FINGER1
CONTACT-LOCATION: (-4.66 -0.50 -9.60)
ACTIVE: T}}

The output of the recognition program in the next experiment (which involves a precision grasp shown in Fig. 47(b)) is:

Effective cohesive index = 0.568797
Virtual finger #1 = VF0 = { 0 }; Cohesive index = 1.0
Virtual finger #2 = VF1 = { 1 2 3 4 }; Cohesive index = 0.647

Contact code is 111110
Grasp is a precision grasp.
N_0 = 5; N_1 = 5

Grasp is Fingertip Grasp.
Grasp is FIVE-FINGERED GRASP.
Number of virtual fingers = 2
Number of opposition spaces = 1
Virtual fingers involved in opposition spaces:
opp0: VF0, VF1; Opposition strength = 1.000

The contact code indicates that all the fingers are active and the palm is inactive. N_0 refers to the total number of active fingers while N_1 refers to the total number of active segments. A precision grasp is classified as a *fingertip grasp* if N_0 = N_1 or a *composite non-volar grasp* if N_0 < N_1. The schemata produced are similar to those shown for the first experiment, except of course for the values of their slots.

5.9 Determining Strength of Opposition between Two Virtual Fingers

Let the two virtual fingers and their real finger compositions be $\mathrm{VF}_1 = \{a_1, ..., a_m\}$ and $\mathrm{VF}_2 = \{b_1, ..., b_n\}$ respectively. The strength of opposition between VF_1 and VF_2 is defined to be

$$S(\mathrm{VF}_1, \mathrm{VF}_2) = \max_{\substack{1 \le i \le m \\ 1 \le j \le n}} \mathrm{m}'_{a_i \to \{1, \ldots, l_{a_i}\}, b_j \to \{1, \ldots, l_{b_j}\}}$$

where $\mathrm{m}'_{i \to \{1, \ldots, l_i\}, j \to \{1, \ldots, l_j\}} =$

$$\frac{1}{2}\left(\prod_{q=1}^{l_j} \max_{p \in \{1, \ldots, l_i\}} \mathrm{m}'_{i \to p, j \to q} + \prod_{p=1}^{l_i} \max_{q \in \{1, \ldots, l_j\}} \mathrm{m}'_{i \to p, j \to q} \right)$$

$$\mathrm{m}'_{ij} = 1 - \frac{\min(|\mathbf{f}_i|, |\mathbf{f}_j|)}{\max(|\mathbf{f}_i|, |\mathbf{f}_j|)} \frac{1 + \mathrm{D_c}(i, j)}{2}$$

and

$$\mathrm{D_c}(i, j) = \frac{\mathbf{f}_i \bullet \mathbf{f}_j}{|\mathbf{f}_i||\mathbf{f}_j|}$$

$i \to p$ denote the pth segment of the ith finger; $\mathbf{f}_i$ and $\mathbf{f}_j$ are two effective normal forces acting on the grasped object due to finger segments i and j. The strength of opposition is at its highest at unity when the two fingers are exerting forces directly against each other.

5.10 Summary of Chapter

A framework for recognizing a grasp has been described in this chapter. A 3-D structure comprising a network of effective contact points of the hand with the grasped object is pro-

posed as a tool for grasp analysis. We call this 3-D structure the *contact web*. It enables the grasp to be classified in a more continuous manner. In addition, by employing a particular real finger to virtual finger mapping, the grasp can be described in higher level conceptual terms such as virtual finger composition and opposition space. Another important consequence of this mapping is an index called the *grasp cohesive index*, which can be used to identify the grasp.

The grasp is actually one of the three identifiable phases in a grasping task. The other two phases are the pregrasp and manipulation phases. The following chapter covers the task analysis, apart from human grasp identication and subsequent to temporal task segmentation.

Chapter 6

Task Recognition Module: Task Analysis

The chapter describes the analyses that can be performed on the observed data of the human task execution. These analyses are subsequent to temporal task segmentation (Chapter 4) and in addition to human grasp recognition (Chapter 5); they are object motion extraction and detection of repetitive motion, such as turning a screw into a threaded hole.

6.1 Organization of Chapter

This chapter begins by listing the primary sources of errors in the observed data – these are the reasons why additional processing is required to compensate for their effects. We have implemented two versions of the observation system. The first version comprises the light-stripe rangefinder, *Cyber*Glove, and Polhemus device, while the second comprises the multibaseline stereo system, *Cyber*Glove, and Polhemus device. The data processing and task analysis of object motion tracking associated with both versions are described and contrasted. The task analysis of repetitive motion detection, which is dependent only on *Cyber*-Glove and Polhemus data, is subsequently described.

6.2 Task Analysis - Sources of Errors

After the task breakpoints have been identified, we can then proceed to identify the grasp (Chapter 5) and extract object motion during the manipulation phase. However, these two processes are complicated by, among others, errors in the Polhemus readings which cause the raw data of the hand position and configuration to be unreliable. The sources of the errors are:

1. *Distortion of magnetic field by nearby ferromagnetic material. This effect is significant because the Polhemus device uses ac magnetic field technology to determine the relative pose of the sensor to the source.*
2. *Inaccuracies in localization of the Polhemus device in the range image during rangefinder-to-Polhemus frame transform calibration. The inaccuracies are in part attributed to the model built in the geometric modeler not exactly corresponding to the actual shape and specified stepsize and resolution of the fitting algorithm.*
3. *Inaccuracies in the modeling of the hand.*
4. *Misalignment of the Polhemus sensor relative to the hand. This happens as the sensor is not rigidly fixed to the glove.*
5. *Misalignment of the Polhemus source relative to the table, since it is not very rigidly affixed to it.*
6. *Inaccuracies in the range data.*

Since a task can be broken down into its constituent subtasks, we can then analyze each subtask individually. The subtask analysis involves grasp identification and object motion extraction; we first illustrate this analysis with experiments involving 1-tasks[1]. The 1-tasks were first recorded using Version 1 of the observation system.

6.3 1-task Analysis Using Version 1 of Data Acquisition System

Analyzing subtasks is equivalent to analyzing 1-tasks, since each 1-task contains a subtask; there is no loss in generality in illustrating the analysis using 1-tasks. Each subtask can be characterized in terms of the grasp used and object motion during the manipulation phase. The grasp can be identified from contact information between the hand and object at the grasp frame identified by the task segmentation algorithm. This is done by mapping the low-level contact information such as the contact position and normal into increasingly abstract entities such as functionally equivalent groups of fingers and the degree of interaction between these groups (Chapter 5). The method to determine the object motion is described in section 6.3.2.

1-task recording was initially done using the first version of the data acquisition system (comprising the light-stripe rangefinder, *Cyber*Glove, and Polhemus device).

6.3.1 Task Segmentation, Grasp Identification and Manipulative Motion Extraction for 1-tasks

A major problem in identifying the grasp is the imperfect positional information obtained from a real data acquisition system such as ours. In addition, the exact moment of grasping cannot be pinpointed due to the discrete sampling of the hand location and configuration. As

1. A 1-task involves only one object manipulation phase.

a result, we have to resort to extra preprocessing to accommodate such data imperfections, specifically adjusting the orientation of the hand at the grasp frame.

The processes of segmenting the task and determining the grasp and manipulative (i.e., object) motions are done using a three-pass approach. The first pass establishes the motion breakpoints while the second pass involves adjusting the pose of the hand and subsequently determining the grasp employed in the 1-task. Finally, the effect of the reorientation of the hand is propagated throughout the 1-task sequence and the object motion is then extracted using the approach delineated in the following subsection. The details of the three-pass approach are as follow:

Pass 1:

1. *Estimate pose of object from the before-task range image.*

   The initial gross position (but not the orientation) of the object of interest is determined by subtracting the 3-D elevation map of the scene after the task from that before the task. The 3DTM[1] program is then used to localize the object. Two refinements were made: (a) use three orthogonal initial poses and pick the final estimated pose with the least RMS fit error; and (b) use coarse-to-fine stepsizes.

2. *Calculate the motion profiles (speed, fingertip polygon area, and volume sweep rate).*
3. *Determine the motion breakpoints from the motion profiles as described earlier.*

Pass 2:

1. *From known motion breakpoints (determined in Pass 1), calculate the object motion associated with the manipulation phase (which is bordered by the grasp and ungrasp transitions).*
2. *At the grasp frame, determine the grasp employed.*

   Due to the errors in the Polhemus and *Cyber*Glove readings, the oriented hand may intersect the object. The hand is reoriented (subject to the fixed position of the Polhemus sensor) until: no interpenetration between the hand and object occurs; and the weighted sum of distances between the hand contact points and the object is minimized.

   The determination of the "optimal" hand pose is done with direct search with rotational increments of 1.15^{o} and limited to a maximum of 60^{o} rotation about discretely sampled axes (80 directions sampled on a once-tesselated icosahedron). (An increment of 1.15^{o} would produce at most an error of about 2 mm at a point 10 cm away from the rotation center.)

---

1. Short for 3-D template matching [153]. See Appendix A for a brief description of this 3-D object localization program.

The object is stored as a collection of oriented surface points (position and normal information associated with each point) whose spacing is typically between 4.0-7.5 mm. This spacing of the object depends on the object size - it is increased for a larger object size. The nearest distance of each hand contact point to the object is then estimated using this oriented point representation.

Once the "optimal" pose of the hand and the object-contact information have been extracted, the grasp is then recognized using the classification scheme described in Chapter 5.

Pass 3:

1. *Propagate adjustment in both distal and proximal motions throughout the task due to hand reorientation in Pass 2.*
2. *The gross after-the-task pose of the object is determined by successively applying the distal transformations in the frames composing the manipulation phase to the original object position (i.e., prior to the task). This after-the-task pose is refined using the 3DTM program.*

6.3.2 Determining Object Motion during Manipulation Phase

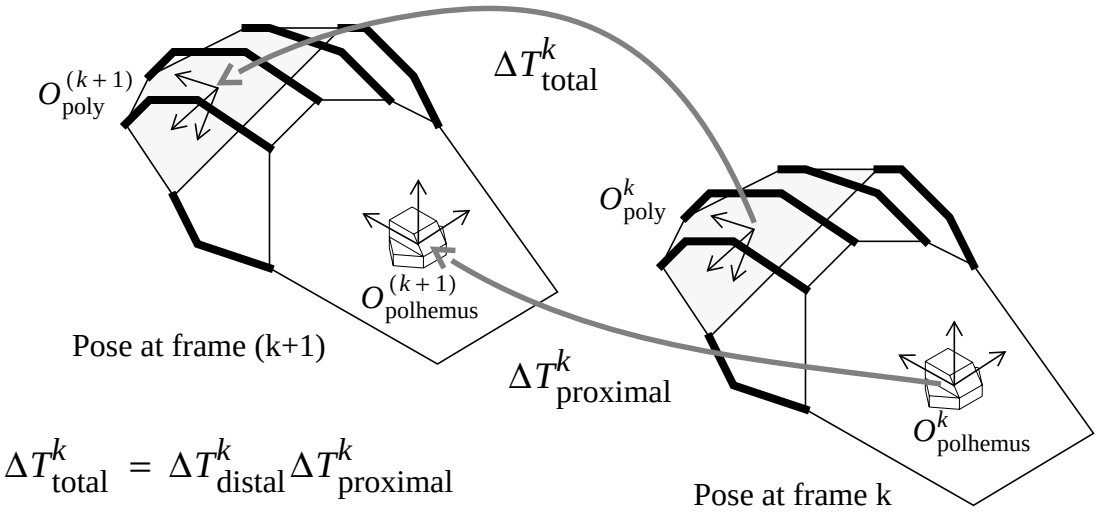


Fig. 48 **Total and proximal motions from frame k to k+1 during manipulation phase**

It may be useful to determine the *proximal motion* (which corresponds to the motion of the arm and wrist) and *distal motion* (which corresponds to motions of the fingers, otherwise referred to as "precision handling" [81]). The *total motion*, which is the overall effect of both the proximal and distal motions, directly yields the object motion. Meanwhile, the proximal and distal motions yield information on which component of the hand/arm motion is contributing to the object motion.

We can determine the object motion transformations (i.e., the total motion) in the manipulation phase once we have identified the task motion breakpoints. Suppose the *k*th frame has been identified as the grasp frame and the *l*th frame the ungrasp frame in the task sequence of *N* frames. The desired object change in pose at frames between *k* and *l* (i.e., during the manipulation phase) can be be determined (Fig. 49) from (5):

$$\Delta T_{k,\,k+j} = T^{k+j}_{hand} \left(T^{k}_{hand}\right)^{-1} \tag{5}$$

where T^{k}_{hand} is the total transform associated with the motion of both the fingers and hand at the *k*th frame.

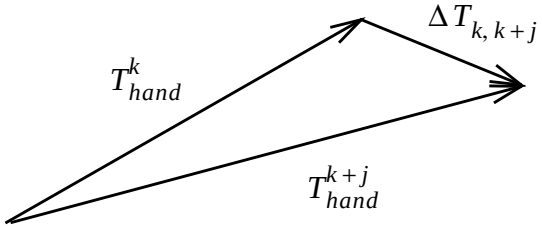


Fig. 49 **Determining the differential motion between two frames in the manipulation phase**

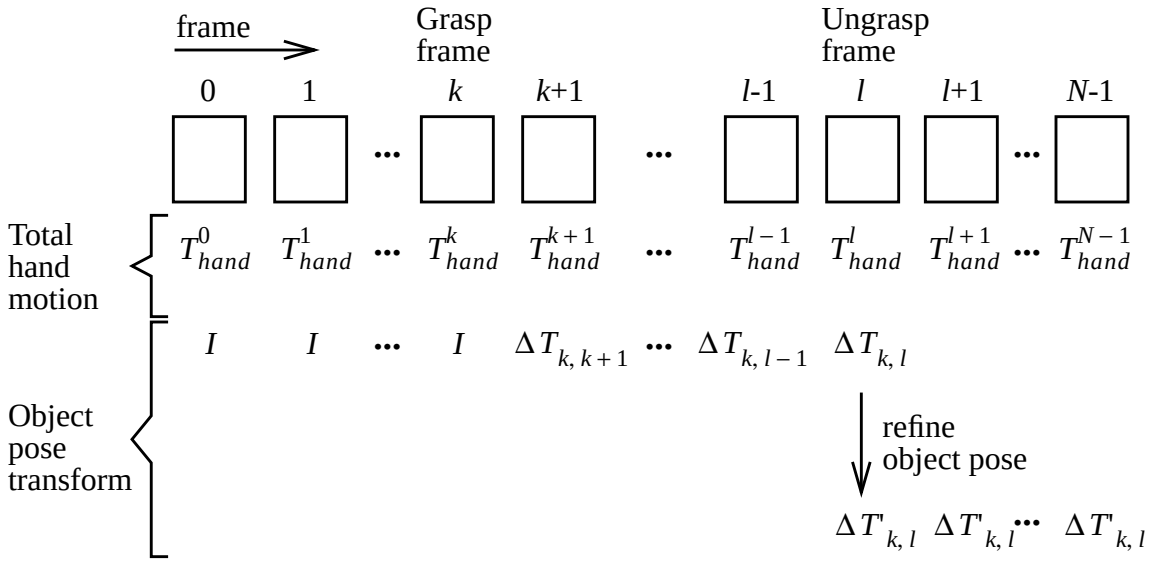


Fig. 50 **Determining pose of object throughout task sequence of N frames**

Based on (5), we can then calculate the object pose transformations at each frame within the manipulation phase as shown in Fig. 50. The pose of the object at the end of the manipulation phase is most likely not very accurate, due to measurement inaccuracies. This pose is refined using a least-squares distance error minimization technique [153].

6.3.3 Results of Applying 3-pass Algorithm

We have applied the 3-pass algorithm on several real 1-tasks to determine the motion breakpoints, identify the grasp employed, and recover the object motion. Two of these are described here. The first 1-task involves picking up a cylinder from one location and placing it on a different location. The results of the first pass are shown in Fig. 51 and Fig. 52. (Note that in Fig. 52, P refers to the pregrasp phase, g the grasp phase, and M the manipulation phase.) The pose of the object prior to the performance of the 1-task has been estimated from the range image. As shown in Fig. 52, the motion breakpoints (grasp and ungrasp points) as well as the pregrasp, manipulation, and depart phases have all been located.

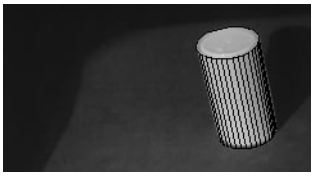


Fig. 51 **Initial pose of the cylinder (1-task #1)**

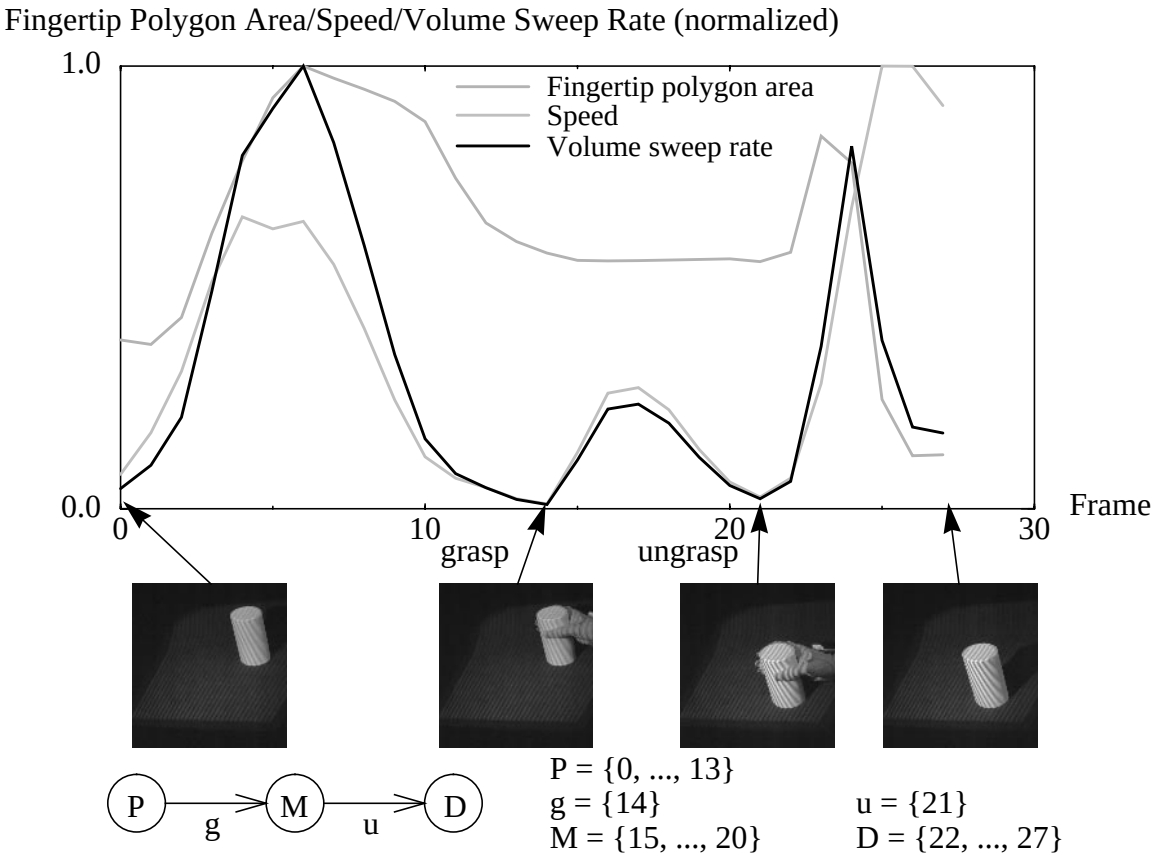


Fig. 52 **Motion profiles and the identified motion breakpoints (1-task #1)**

It is interesting to note the profiles of the average joint flexion angles for each finger and all the fingers (Fig. 53(a)) for this 1-task. As expected, there was very little change in the average flexion angles during the manipulation phase, during which the cylinder was grasped with a power cylindrical grasp. It is also interesting to note that profile of the reciprocal of the average joint angles for all the fingers (Fig. 53(b)) is very similar to that of the fingertip polygon area. This suggests that this may be another metric that can be used in the task segmentation scheme.

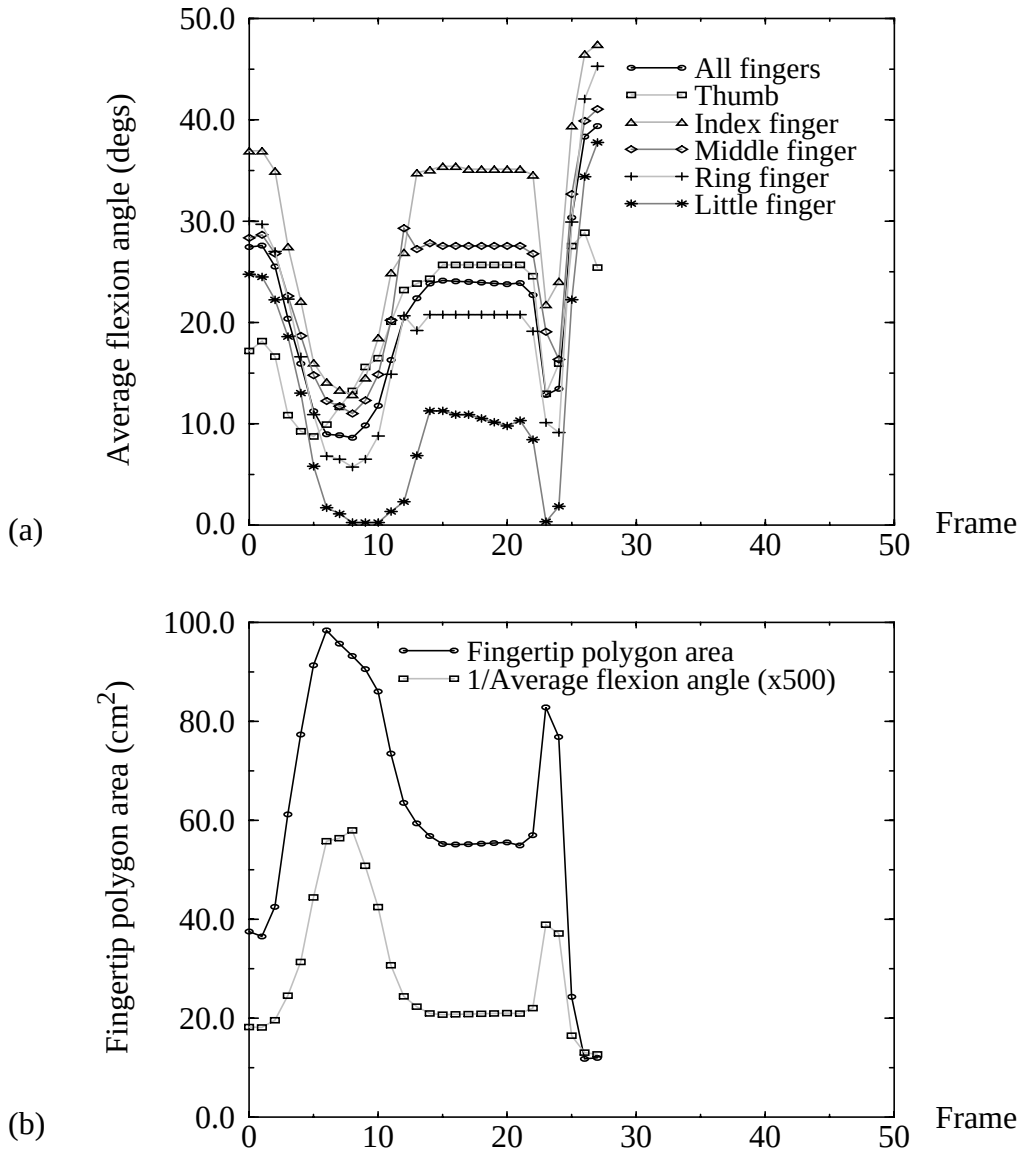


Fig. 53 **Average flexion angle profiles (1-task #1): (a) each and all fingers; (b) comparing the scaled inverse average angle to the fingertip polygon area.**

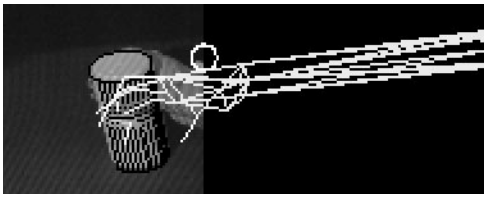
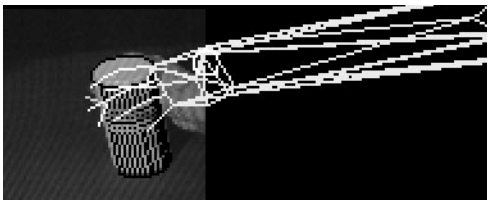
(a)  (b) 

Fig. 54 **Reorienting the grasp in Pass 2: (a) initial pose of the hand relative to the object; (b) final pose of hand relative to cylinder**

Once the hand was reoriented (Fig. 54), the grasp was then correctly identified as a type 2 'coal-hammer' cylindrical grasp[1] using the grasp classification scheme described in Chapter 5. By propagating the extracted object motion during the manipulation phase, the object pose was then estimated (Fig. 55(a)). The pose is subsequently refined (Fig. 55(b)).

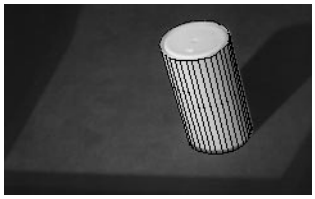
(a)  (b) 

Fig. 55 **Pose of the cylinder after the task subsequent to Pass 3: (a) pose obtained by successively applying total motion transformations in the manipulation phase; (b) refined pose using the 3DTM program [153].**

The second 1-task considered is picking up a stick and inserting it through a hole in a castle-shaped object. The two objects involved in this 1-task and the superimposed model of the stick are shown in Fig. 56. Fig. 57 depicts the extracted motion breakpoints and phases of this task.

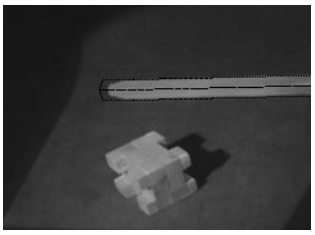


Fig. 56 **Initial pose of the stick (1-task #2)**

As in 1-task #1, the shape of the reciprocal of the average angle profile closely resembles that of the fingertip polygon area (Fig. 58(b)). However, because the stick was held in a precision grasp and the object motion was a combination of translational and rotational motions, there were changes in the average finger joint angles during the manipulation phase (as evidenced in Fig. 58(a)).

1. A 'coal-hammer' cylindrical grasp is one in which the thumb is highly abducted (i.e., significantly deviated from the plane of the palm). This 'coal-hammer' cylindrical grasp is of type 2 because the thumb touches the object. See Chapter 5 for more details.

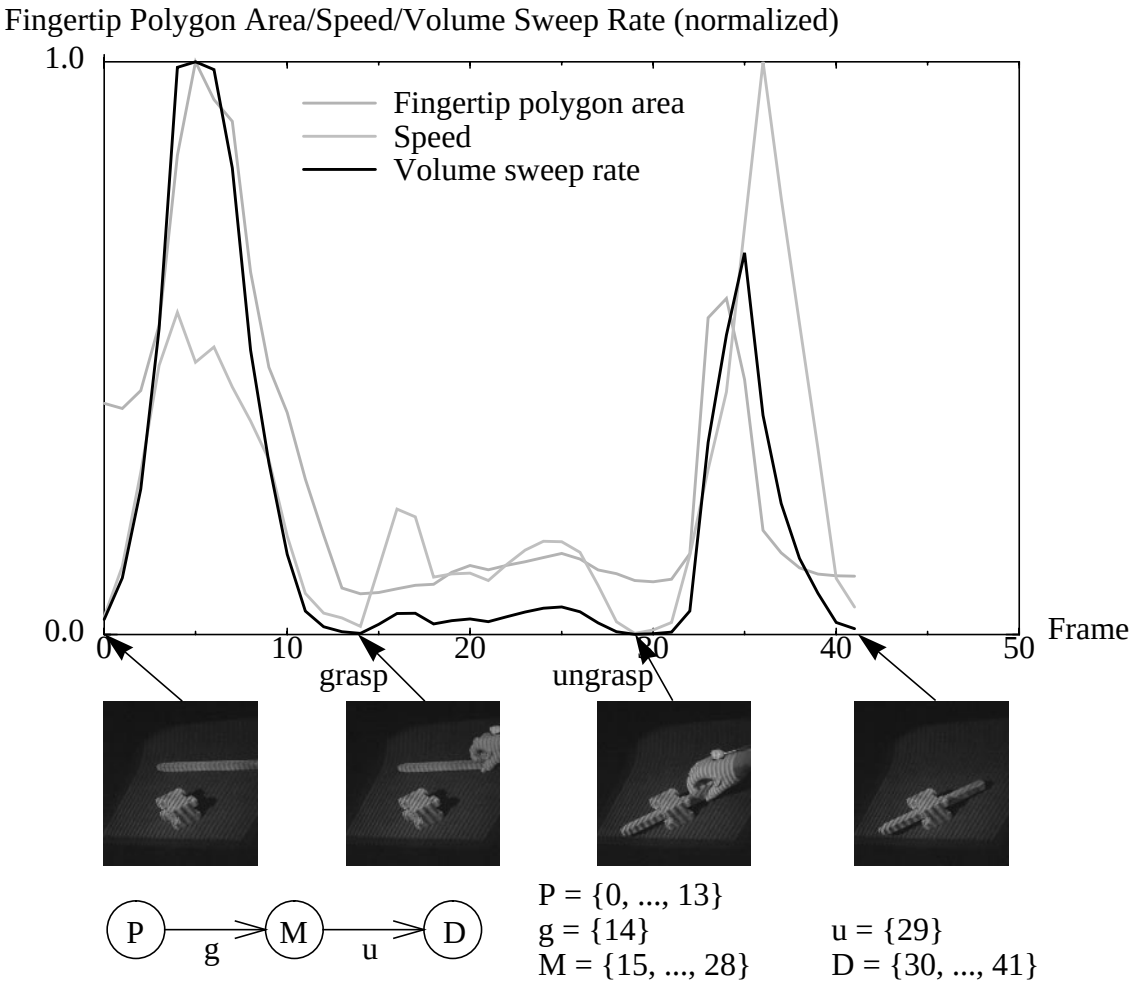


Fig. 57 **Motion profiles and the identified motion breakpoints (1-task #2)**

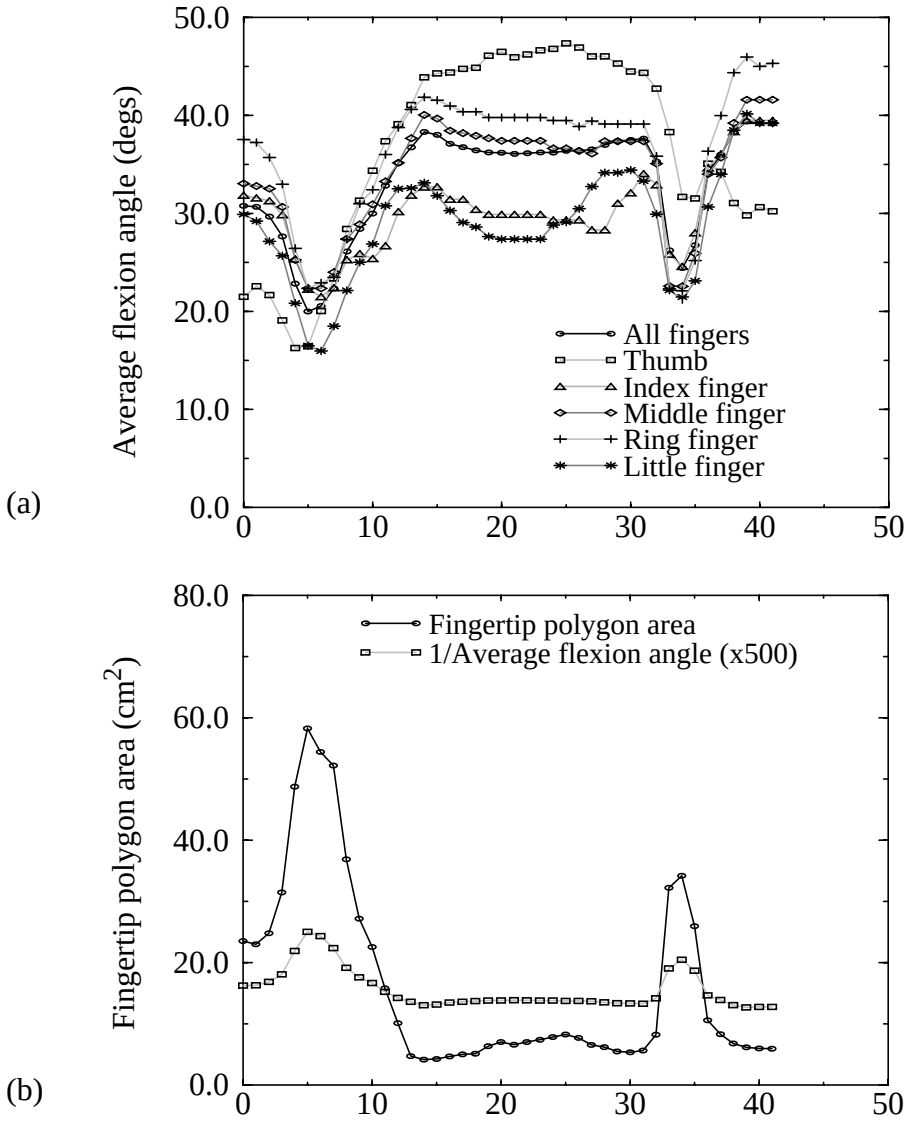


Fig. 58 **Average flexion angle profiles (1-task #2): (a) each and all fingers; (b) comparing the scaled inverse average angle to the fingertip polygon area.**

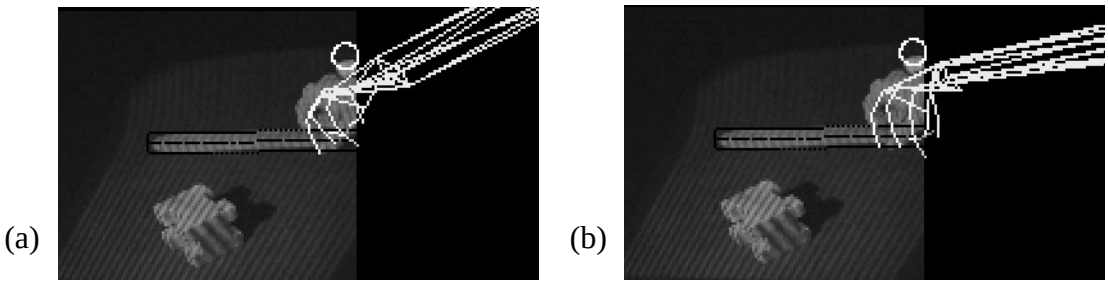


Fig. 59 **Reorienting the grasp in Pass 2: (a) initial pose of the hand relative to the object; (b) final pose of hand relative to stick**

Fig. 59 shows the pose of the hand relative to the stick at the grasp frame before and after reorientation. The grasp was identified as a precision grasp. However, because the middle

segments of the four fingers are within the tolerance range of the object (which is set at 1.0 cm), the grasp is classified as a composite nonvolar grasp (Chapter 5), specifically a prismatic pinch grasp. The grasp that was actually employed in the task is a five-fingered prismatic precision grasp; it can be seen from this result that the while the general grasp classification is correct, the specific category is sensitive to orientation and position errors.

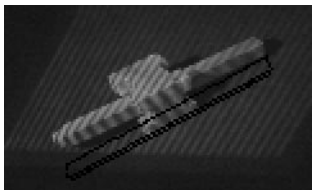
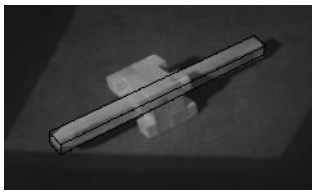
(a)  (b) 

Fig. 60 **Pose of the stick after the task subsequent to Pass 3: (a) pose obtained by successively applying total motion transformations in the manipulation phase; (b) refined pose using the 3DTM program [153].**

Fig. 60(a) shows the estimated object pose at the end of the task from extracted total motion, while Fig. 60(b) shows the refined final object pose.

6.4 Other Possible Input Modalities

The variations in the fingertip polygon area during manipulation are due to both the fingertips rolling on the object surface and noisy data. For example, in 1-task #2 (Fig. 57), there is appreciable rolling of fingers on the object surface when rotating the stick, causing the variation of the fingertip polygon area during manipulation. In 1-task #1 (Fig. 52), the variation is probably mostly due to noisy data (though it is difficult to keep the fingers still during direct object translation).

As mentioned in section 6.2, the Polhemus 3-D tracker, which uses an ac current emitter, is susceptible to interference due to nearby ferromagnetic material. The interference effect is rather significant in our setup, contributing to the errors. The most simplistic way to reduce this is to clear the workspace of nearby metallic material; apart from that, one can use the Ascension Bird 3-D tracker. This tracker uses a dc emitter and minimizes the effect of conductive metals, though, it is still affected by ferromagnetic material [107].

If the *Cyber*Glove had tactile sensing, the location of the contact points would very likely be easier to calculate, but the error in hand pose would still have to be accounted for. The presence of tactile sensors would definitely help in the temporal task segmentation; the breakpoints would correspond to the sudden changes in the outputs of the tactile sensors during the execution of the task.

Error minimization may be achieved using dense range data – we have developed an active 4-camera multibaseline stereo system capable of video rate intensity image acquisition [74] for this purpose. Using this stereo system would enable us to extract depth data at every

frame, instead of just those just prior to and just after the execution of the task as reported in this paper. Results have shown that the average errors in fitting planes and cylinders to the stereo depth data are less than 1 mm at distances of 1.5-3.5 m away from the camera setup.

6.5 Task Analysis Using Version 2 of Data Acquisition System

The second version of the observation system includes a multibaseline stereo system that is able to capture images rapidly. When interlaced with data sampling of the *Cyber*Glove and Polhemus device, the overall rate is about 4.5-5 Hz. As a result, we are able to record tasks with more than just one object manipulation in a single sequence. This is in contrast to the first version of the observation system that is able to capture task sequences that contain only one object manipulation. As a result of available depth data at each frame during the task sequence, the object can be tracked more reliably than just using *Cyber*Glove and Polhemus data.

Since there may be more than one object manipulation in the task sequence, additional processing has to be performed to identify which object has been manipulated at each manipulation phase.

6.5.1 Identifying the Grasped Object

The objects in the scene are first manually localized by the user using an interface that features both 2-D images and 3-D scene points as shown in Fig. 61. Once this is done, the program automatically finetunes the localization of the object poses (the results of which are shown in Fig. 61).

(a) (b)

Fig. 61 **Object model fitting: (a) Superimposed object models (cylinder and box) in a scene, and (b) 3-D views of the scene**

In a given task which involves more than one object manipulation and a scene with more than one object, we have to somehow identify the object that is being grasped during each manipulation phase. One direct way would be to preprogram the list of objects that are manipulated in the task. The more automatic solution, which is used here, is to infer from range data the identity of the object being grasped and manipulated during each subtask. The approach is a simple one, and is based on the proximity of the fingertip polygon centroid of the human hand to the objects in the scene during the grasp phase. The smallest distance of each object to the fingertip polygon centroid is first calculated. The object whose closest distance is minimum is deemed to be the grasped object. Object motion during the subsequent manipulation phase affects only the identified object.

6.5.2 Tracking 3-D Object

Once each object has been identified as the object being moved in each manipulation phase, we can then proceed to track the object at each manipulation phase. It may not be possible all the time to track the object using just the 3DTM algorithm (pose refinement program) at the next frame just based on the object pose at the present frame. This is because significant displacements may have occurred between frames, and since the 3DTM algorithm is basically a local one, the wrong pose may be extracted.

Using *Cyber*Glove and Polhemus Data as First Approximation

Because we require reasonable starting poses as input to the 3DTM program, we use the recorded *Cyber*Glove and Polhemus data as an approximation. The 3-D object tracking scheme (for one object manipulation for clarity) is depicted in Fig. 62. Contrast this scheme to that shown in Fig. 50.

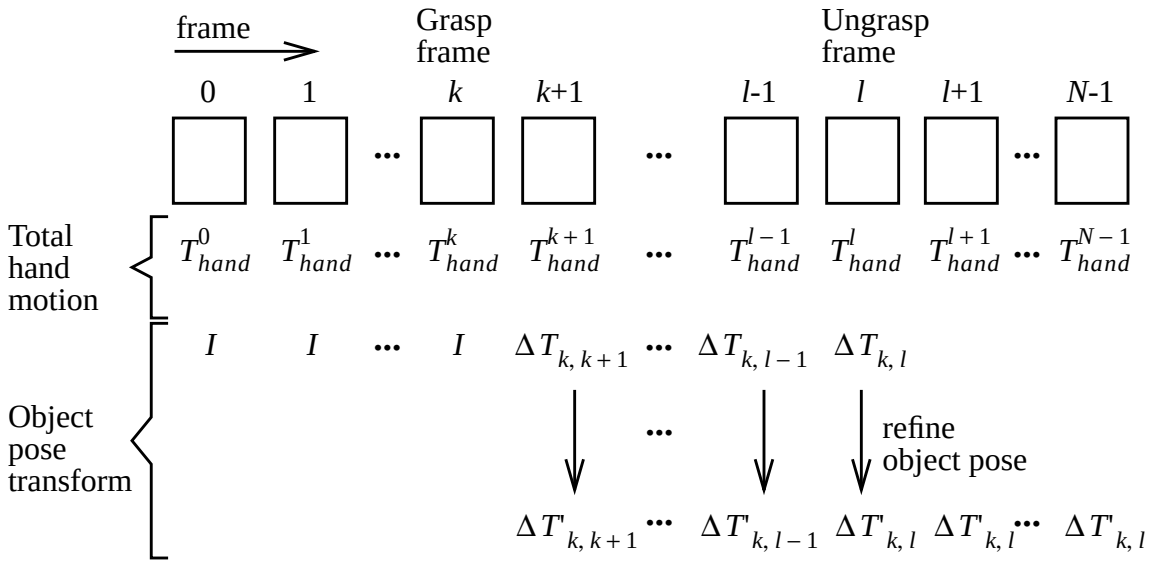
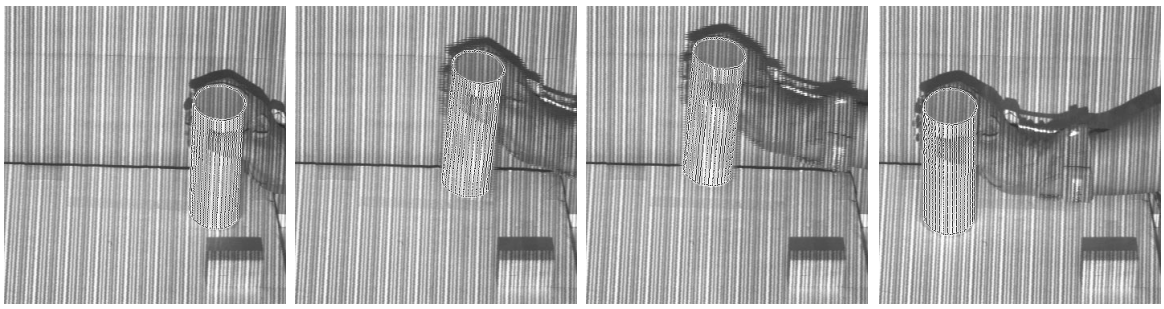


Fig. 62 **Determining pose of object throughout task sequence of N frames**

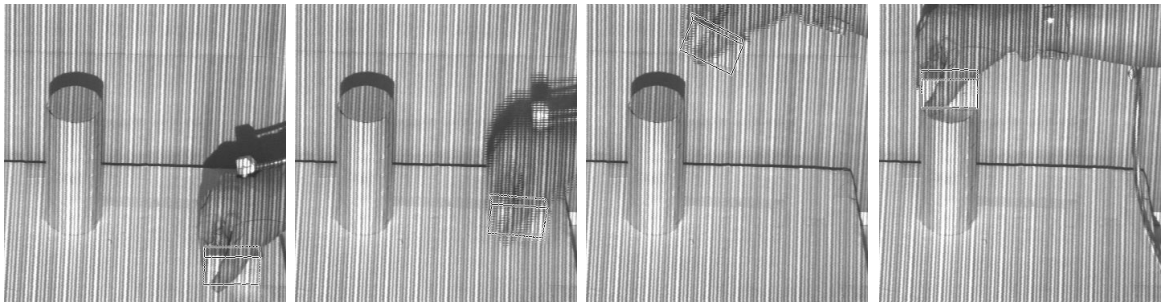
At each frame during each manipulation phase, the gross location of the 3-D object is given by the Polhemus data. The object pose is subsequently refined using the 3DTM program. An example of object tracking is shown in Fig. 63. The task involved picking and placing a cylinder followed by picking a box and placing it on top of the cylinder. Prior to this, these two manipulation phases have been detected by the temporal task segmentation routine (the results of which are shown in Fig. 18 in Chapter 4).

Object manipulation #1



Frame 8 Frame 11 Frame 12 Frame 15

Object manipulation #2



Frame 24 Frame 26 Frame 28 Frame 31

Fig. 63 **3-D object tracking example**

Direct tracking in this manner works if there is little or no object occlusion. Additional processing has to be added to tackle the problem of significant occlusion between objects.

Handling Significant Occlusion Between Known Objects

While the 3DTM program is robust to some degree of object occlusion, significant occlusion may occur in the course of the task performance, as can be seen in Fig. 65. In this example, the task is to pick and insert a peg into a receptacle. In cases such as this, the program fails. The following steps were employed to deal with this problem:

1. *Masking out range pixels corresponding to stationary objects of known poses*

   This is done to minimize interference in object localization. An example result of this operation is shown in Fig. 64, in which the receptacle is "blanked out" while the pose of the peg is being localized.

Fig. 64 **Resulting of masking the stationary object and fitting the moved object**

2. *Detecting and avoiding object collision*

   Despite the masking operation, nearby range data may still interfere with the pose localization. In addition, we want to be certain that the object being moved does not collide with any stationary objects in the scene, to avoid problems with actual robot execution.

   We use a simple collision detection and avoidance scheme to both aid in localization and to ensure that objects do not interpenetrate at any time during task execution. Each object is represented by surface points of a predetermined spacing (of about 2.5 mm apart). Corners and edges are definitely sampled. Associated with each surface points is the object normal at that point. The normal at an edge or a corner is calculated to be the mean of the normals at adjacent faces. With these information, we can determine to a reasonably good approximation, if collision occurs, and where.

   The object collision avoidance algorithm is as follows: When the object being moved ("non-stationary object") and a stationary object interpenetrates at some area, the points at these areas "pushes" non-stationary object away in the direction of the normals at the affected points of the stationary object. The repelling motion is the net twist acting on the centroid of the non-stationary object. The repelling motion formulation is similar to that in the adjustment phase (item 2) for power grasp generation described in Section 7.7.3. Unlike the adjustment phase, there are no attractive forces between objects.

   The object collision detection and avoidance scheme has been incorporated into the iterative pose localization of 3DTM. A result is shown in Fig. 65.

(a) (b)

Fig. 65 **Result of detecting and avoiding object collision in pose localization: (a) Using just 3DTM without collision detection; (b) 3DTM with collision detection. Note that in both cases, stationary object masking was performed.**

6.5.3 Repositioning Hand for Grasp Recognition

As with the first version of the observation system, the pose of the hand is not exact due to inaccuracies of the Polhemus data. The manner in which hand adjustment is made prior to grasp identication is the same as described in section 6.3.1 (Pass 2, item 2). A result of hand adjustment for the task involving pick-and-place of a cylinder and box is shown in Fig. 66.

(a) (b)

Fig. 66 **Hand adjustment example: (a) Before, and (b) after adjustment.**

6.6 Comparison Between Two Versions

As can be seen, the kind and amount of processing required to generate task description is extremely dependent on the available sensors in the observation system. For the first version of the observation system that comprises the light-stripe rangefinder, *Cyber*Glove, and Polhemus device, there is a limitation of one object manipulation per task sequence. In addition, object motion is directly extracted from the *Cyber*Glove (through the hand model) and Polhemus device. This is subject to both errors in the hand model as well as the Polhemus device.

On the other hand, for the second version of the observation system that comprises the active multibaseline stereo system, *Cyber*Glove, and Polhemus device, there is no limit to the number of object manipulations within the task sequence. The restriction is only due to the local iWarp memory of about 500 MBytes, which restricts each sequence to about 500

frames. Because stereo range data are also available at each frame, object motion can be more reliably extracted from this stream of perceptual data.

Another useful task analysis that can be performed is the detection of repetitive motion in the task sequence. It is usual for assembly tasks to include screw or bolt fastening operations – these are repetitive motions that are manifested as regular patterns in the human hand motion profiles. Examples of these can be seen in Fig. 16 and Fig. 17 in Chapter 4. Detection and localization of such activities is made possible by using local Fourier, or spectrogram, analysis.

6.7 Detection and Localization of Repetitive Motion using Spectrogram

The spectrogram of a signal is a space/frequency representation which comprises a series of small-support, Fourier transforms of the signal, each centered around a different point of the signal [77]. For a 1-D signal, this space/frequency representation is 2-D. It reveals the frequency content of the signal within the vicinity of each different point. By determining the frequency content locally at each point, we can localize the signal having a particular maximum instantaneous frequency.

It is obvious from the nature of the spectrogram of its utility in detecting and localizing repetitive motion within the manipulation phase. The motion that is most frequently associated with repetitive motion is the screwing motion. We detect the repetitive motion by analyzing the spectrogram of the fingertip polygon area profile throughout the task, with the following conditions:

1. *Ignore low frequencies.*

   The dc component as well as the first non-zero frequency component are ignored.

2. *Consider only parts of the spectrogram that are associated with manipulation phases.*
3. *Find the highest frequency peak at each point within a manipulation phase.*

   This peak has to be higher than a calculated minimum magnitude determined to the average magnitude of the entire spectrum at that point. In addition, this peak has to be associated with a frequency higher than a determined minimum frequency. This minimum frequency is calculated based on the assumption that there has to be at least three spatial peaks within the manipulation phase for the establishment of a repetitive action.

   If the duration of the manipulation phase (within which the point lies) in the task is M frames, then the minimum frequency is

$$F_{min} = \frac{P_{min}}{M}$$

where P_{min} is the minimum number of peaks within a manipulation phase (3 in our case).

(a) (b)

Fig. 67 **Spectrogram of a 1-task involving screw-turning actions: (a) top view (b) oblique view. The marks on the spectrogram indicate significant frequency peaks**

Fig. 67 and Fig. 68 show the spectrogram of two different tasks which involve screw-turning actions. The width of the spectrogram window is 19 frames; the duration of each frame is about 0.5 sec. and the maximum frequency detected in the spectrogram is about 1 Hz. As can be seen, the detected peaks cluttered along a line do indicate the existence of such repetitive movements. In spectrograms of other tasks which do not involve repetitive actions, no prominent peaks were detected, as to be expected.

(a) (b)

Fig. 68 **Spectrogram of a 4-task involving a manipulation phase with screw-turning actions: (a) top view (b) oblique view. The marks on the spectrogram indicate significant frequency peaks**

Fig. 69 **Spectrograms of 3 tasks with no repetitive actions.**

6.8 Summary of Chapter

This chapter describes the types of analyses that can be performed on the perceptual data input to characterize the observed task. In addition to task segmentation and grasp recognition detailed in previous chapters, the operations that can be done are extracting object motion and detecting repetitive motion.

The processing involved for the two versions of the observation system are shown and contrasted. It is apparent that the second version provides a richer set of perceptual data in comparison to that of the first, since the second version records (indirectly) the depth information of the scene at every frame during human task execution, in addition to the *Cyber*Glove and Polhemus data. This results in more reliable object motion extraction, and eliminates the limit to the number of object manipulations per task sequence (subject to memory constraints).

Another important operation that can be performed to characterize the task is the detection of repetitive motion, such as the action of turning a screw into a threaded hole. This is accomplished by using spectrogram, i.e., local Fourier, analysis on the hand motion profiles during the manipulation phases.

Finally, it may be useful to characterize the manipulation phase in terms of *total motion* (due to both finger and hand motions), *distal motion* (due to just finger motion) and *proximal motion* (due to just hand motion). While the total motion directly yields the object motion, the proximal and distal motions yield information on which component of the hand/arm motion is contributing to the object motion. This chapter indicates how these motion transformations can be determined.

Chapter 7

Task Translator and Robot System

Once the observed perceptual stream of data has been analyzed and the human actions characterized, this derived information can then be automatically translated into equivalent robot actions. This chapter shows how human grasp description can be used to map the human grasp to that of the manipulator. We are interested in using human cues as hints in planning manipulator grasps and paths.

7.1 Organization of Chapter

This chapter starts with a brief discussion on grasp planning issues. The general approach of planning manipulator grasps from human grasp description is delineated next. Grasp planning depends on the type of grasp used – both approaches of planning power grasps and fingertip precision grasps are given in this chapter. Example grasp mappings are shown. The robot system used as the testbed for proof of concept is also described here.

A major component in task planning is planning the manipulator grasp. In the following section, we briefly discuss some of the issues involved in grasp planning and reiterate the justification for our approach.

7.2 Grasp Planning

Grasp planning involves determining the hand placement relative to the object of interest as well as the posture of the hand assumed in grasping the object, both in a manner consistent to the requirements of the task. It is computationally intensive due to both the large search space and the incorporation of geometric and task-oriented constraints. Globally optimal grasp planning using comprehensive or exhausive search is not feasible, especially with a high number of joints of the robot arm and hand.

Approaches to grasp planning can be categorized as geometric and mechanical [121]. The geometric approach considers only the spatial aspects of the objects and arm/hand system,

while the mechanical approach incorporates the arm/hand kinematics, system dynamics, and their task-related constraints.

The search space for the geometrical approach is multidimensional; for a hand with n_h configuration parameters, the dimension of this search space is $(6+n_h)$, 6 being attributed to the pose of the hand. For a relatively simple manipulator such as the parallel-jaw gripper, the number of potential grasps for a polyhedral object with n_f faces is $O(n_f^2)$. This complexity is compounded by the need to determine the precise location of the contact points or areas among the theoretically infinite number of potential ones for each potential grasp (assuming that the potential grasp is determined to be geometrically feasible). Much of work on grasp synthesis tend to use examples featuring grippers with a small DOF or planar objects. An excellent survey on grasp planning can be found in Pertin-Troccaz's paper [121].

The search space for an optimal grasp can be reduced in size or dimensionality by using certain heuristics or structural constraints. For a 2-finger gripper, for example, the surface element pair selection can be guided by a series of constraints, such as face parallelism, permissible face separation, and non-zero overlap between projected faces [157]. By imposing fixed contact locations or fixed types of grasp, Pollard computes a 6D projection of the 15 DOF configuration space for the Salisbury hand [122]. This reduced space represents the space of wrist poses for geometrically feasible grasps, and is used to search for a globally optimal grasp. In addition, a popular strategy for reducing the complexity of planning the dextrous hand grasp is by using a small set of predetermined grasps or grasping behaviors (e.g., [18], [42] and [126]).

In our approach of robot programming by human demonstration, grasp planning is made more tractable by using cues extracted from human execution of the grasp. Subsequent to temporal segmentation of the recorded task, the human grasp is recognized and the approximate grasp contact locations recovered. This is followed by determining the initial approximate grasp which satisfies kinematic and geometric constraints, and subsequently performing local optimization of the grasp using the approximate grasp as the starting condition as well as a task-oriented objective function. Our grasp mapping approach is described in detail in the following sections.

7.3 Mapping Grasps

7.3.1 General Approach

Planning the manipulator grasp based upon the observed human grasp comprises the following steps:

1. *Local functional mapping*

   This is done by observing the type of grasp used by the human, and the functions and grouping of the palm and individual fingers. The latter is determined using the real finger-to-virtual finger mapping described in Chapter 5. The mapping of the manipulator end-effector to the virtual fingers is dependent on the effective cohesive index and the cohesive indices of the individual virtual fingers.

2. *Adjustable or gross physical mapping*

   This operation is carried out using the outcome of the local functional mapping and the initial location of the manipulator (near the pose of the human hand while carrying out the task). It results in an approximate form of the manipulator grasp of the object which is geometrically feasible.

3. *Fine-tuning or local adjustment of grasp*

   The grasp determined from steps 1 and 2 may not exhibit static stability or that it may not be locally optimal (according to a chosen task-related criterion). By analyzing the criterion and iteratively perturbing the initial grasp configuration, we arrive at a locally optimal grasp. In the case of the power grasp which involve high kinematic constraint, this step is skipped.

Step 1 corresponds to grasp planning at the *functional* level whereas steps 2 and 3 correspond to grasp planning at the *physical* level.

In order to provide both a means for locally adjusting the grasp and for checking the appropriateness of the resulting manipulator grasp, we make use of certain grasp analytic measures.

7.3.2 Analytic Measures of a Grasp

Cutkosky and Howe [30] summarizes a number of analytic measures that have been used by various researchers in their analyses of grasps, several of which are for the purpose of grasp synthesis. We can group these grasp analytic measures into different categories according to both the depth of knowledge of the grasp and hand mechanism, and the type of analysis (Fig. 70). These analytic grasp measures are instrumental in determining the manipulator grasp.

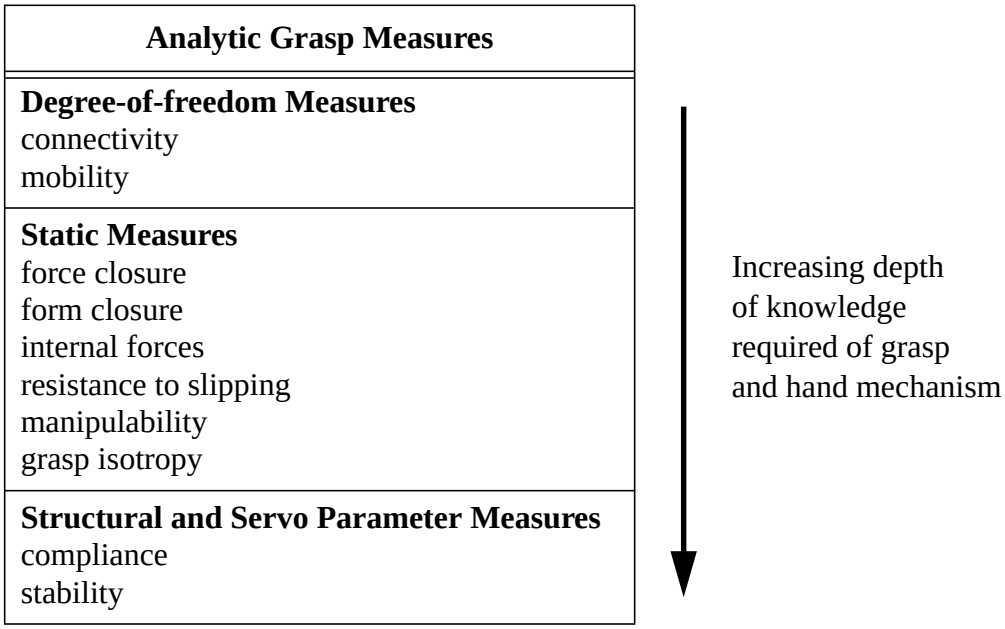


Fig. 70 **Categories of analytic grasp measures**

The overall strategy of mapping grasps is depicted in Fig. 71. The human grasp can be described in a hierarchical fashion. With *a priori* knowledge of the human hand and data derived from observation, certain analytic grasp measures can be extracted (such as connectivity and mobility). Armed with these information, together with *a priori* kinematic, structural and geometric knowledge of the manipulator, the grasp translator would then attempt to produce a manipulator grasp with compatible analytic grasp measures. In this thesis work, the manipulability measure is used as the criterion used in generating the locally optimal manipulator grasp.

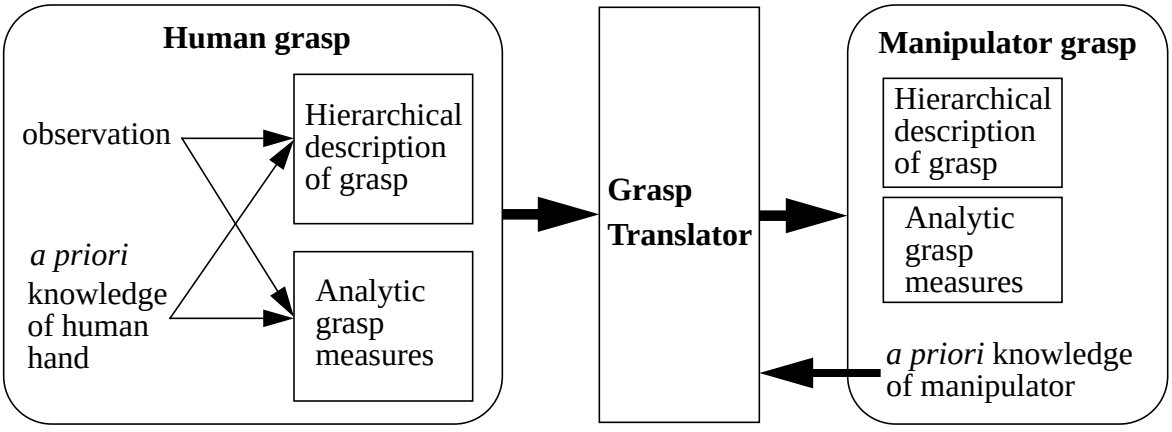


Fig. 71 **Grasp mapping strategy**

The functions of the hierarchical description of the human grasp are:

1. *Local functional mapping; and*
2. *Gross physical mapping.*

The functions of the analytic grasp measures are:

1. *Verification of appropriateness of gross physical mapping; and*
2. *Local fine-tuning or perturbation of grasp to a locally optimal posture.*

7.4 Local Functional Mapping

This first step in mapping grasps is common to all types of grasps, i.e., power and precision grasps. The local functional mapping associates the manipulator fingers to those of the hand, and is generally different for different grasps. To guide the local functional mapping, we first assign manipulator fingers as either a primary finger, secondary finger, or a palm.

7.4.1 Assigning Manipulator Fingers

Prior to mapping the manipulator fingers to those of the human in the observed grasp, the manipulator finger has to be assigned as one of the groups of fingers:

- Primary finger/s

  A finger in this group would correspond to the human thumb, which has at least the same number of degrees of freedom than the other fingers. It serves not only to secure the hold of the object against the palm and/or other fingers, but also manipulate object.

- Secondary finger/s

  This group would correspond to the fingers of the hand other than the thumb. This group of fingers would serve to secure the hold of the object against the palm and/or primary finger/s.

A special case of a “finger” which is also assigned is the palm; it is passive with no local degree of freedom (i.e., discounting the global 6 DOF of the manipulator).

Manipulators are designed with specific functions or level of dexterity in handling objects. The repertoire of grasping abililties may be as limited and simple as just clamping the object between its jaws as a means of securing a hold on it (as in the parallel-jaw gripper). The manipulator may be anthropomorphic to some degree and be able to mimic certain grasps of the human hand; examples of such hands are the Utah/MIT hand [61], Salisbury hand [101], Okada hand [116], Minnesota hand [86], and the Southampton hand [28]. It is usually simple to label *a priori* each of the manipulator fingers as either a primary or secondary finger. It is also conceivable that the fingers can also be classified by comparing the similarity in shape and orientation of the velocity or manipulability ellipsoids of the fingers with the hand

assuming a certain pose (for example, at “mid-posture,” with all the finger joint angles midway between the joint limits), as can be seen in Fig. 72. We chose to specify the type of finger *a priori*.

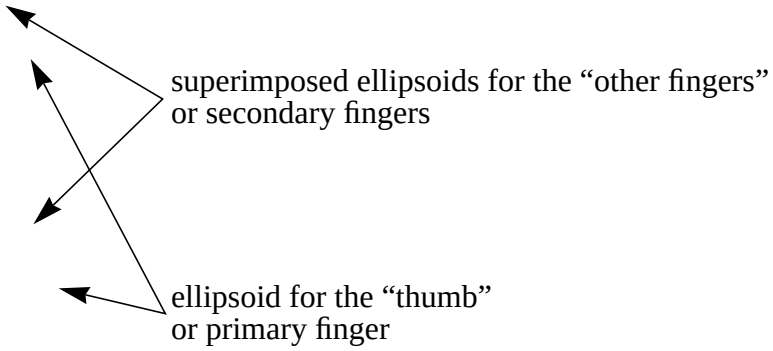


Fig. 72 **Side views of velocity ellipsoids for the Utah/MIT fingers (top) and Salisbury fingers (bottom)**

Once the primary and secondary fingers have been identified, the hand-to-manipulator finger mapping can proceed with the aid of another type of mapping. This mapping relates real fingers to groups of functionally equivalent fingers called *virtual fingers*. A consequence of this real finger-to-virtual finger mapping is the *cohesive index*, which has been explained in section 5.6.1. In short, the cohesive index is associated with the degree to which the real fingers in a virtual finger act in a similar manner; this is determined by comparing the object normals at the contact points. A cohesive index of a virtual finger is maximum at unity, and would indicate that the fingers within that virtual finger act exactly alike.

7.4.2 Using the Cohesive Index to Guide Mapping of Fingers

An indication of how the cohesive index is used to guide the virtual finger mapping between those of the human hand and the manipulator is illustrated in Fig. 73. (VF_n refers to the *n*th virtual finger with C_n being its associated cohesive index.) The palms, primary fingers, and secondary fingers are mapped appropriately as seen in Fig. 73. The cohesive index serves as a check – the virtual finger corresponding to the group of secondary fingers normally has a cohesive index less than that of the collection of primary fingers. The virtual finger composition determined from the real finger-to-virtual finger mapping take precedence, however.

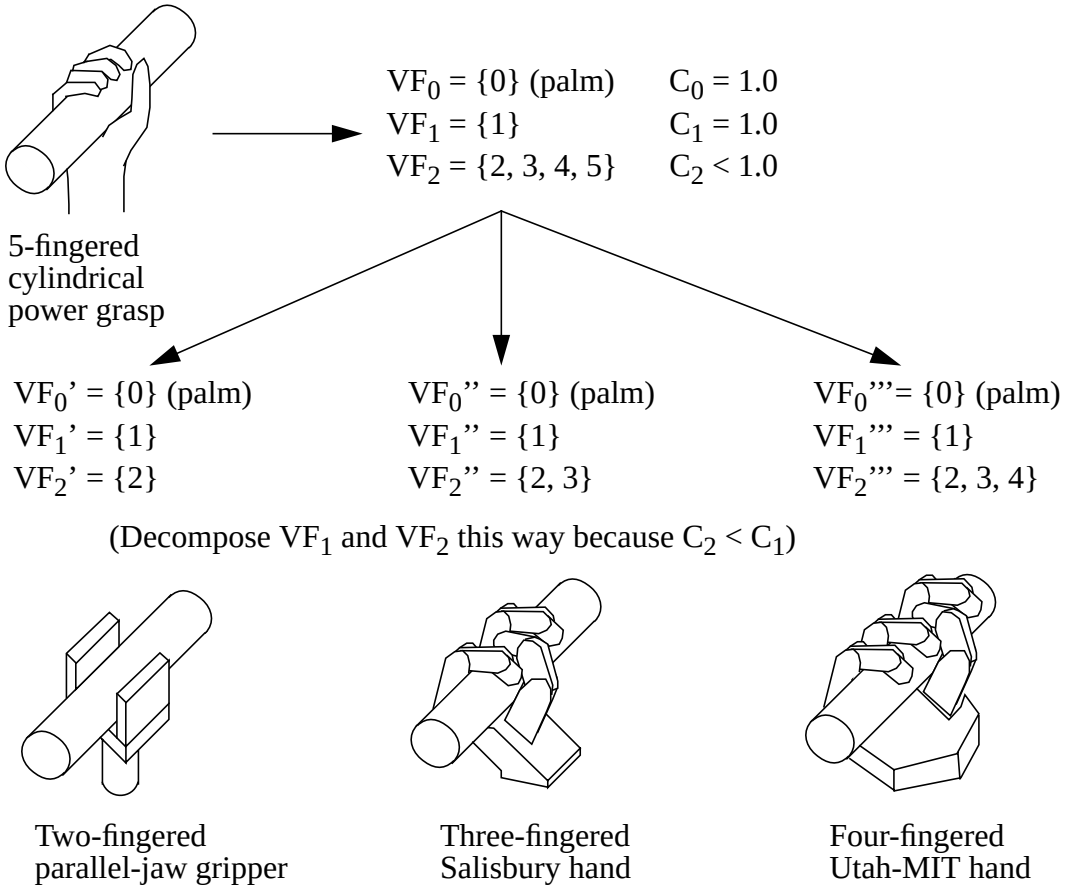


Fig. 73 **Mapping grasps according to cohesive indices of virtual fingers**

Once the functional finger assignments have been accomplished, the more detailed manipulator grasp planning can then proceed.

7.5 Approach in Generating Grasp and Pregrasp Trajectory

The diagram representing the entire manipulator grasp and trajectory planning approach is shown in Fig. 74. Prior to the more detailed manipulator grasp planning, the manipulator assumes a fixed pose and posture relative to the human hand throughout the task execution; local planning is carried out within the vicinity of this pose and posture. Note that by virtue of the different form and function of the power and fingertip precision grasps, they are planned in a different manner elucidated in subsequent sections.

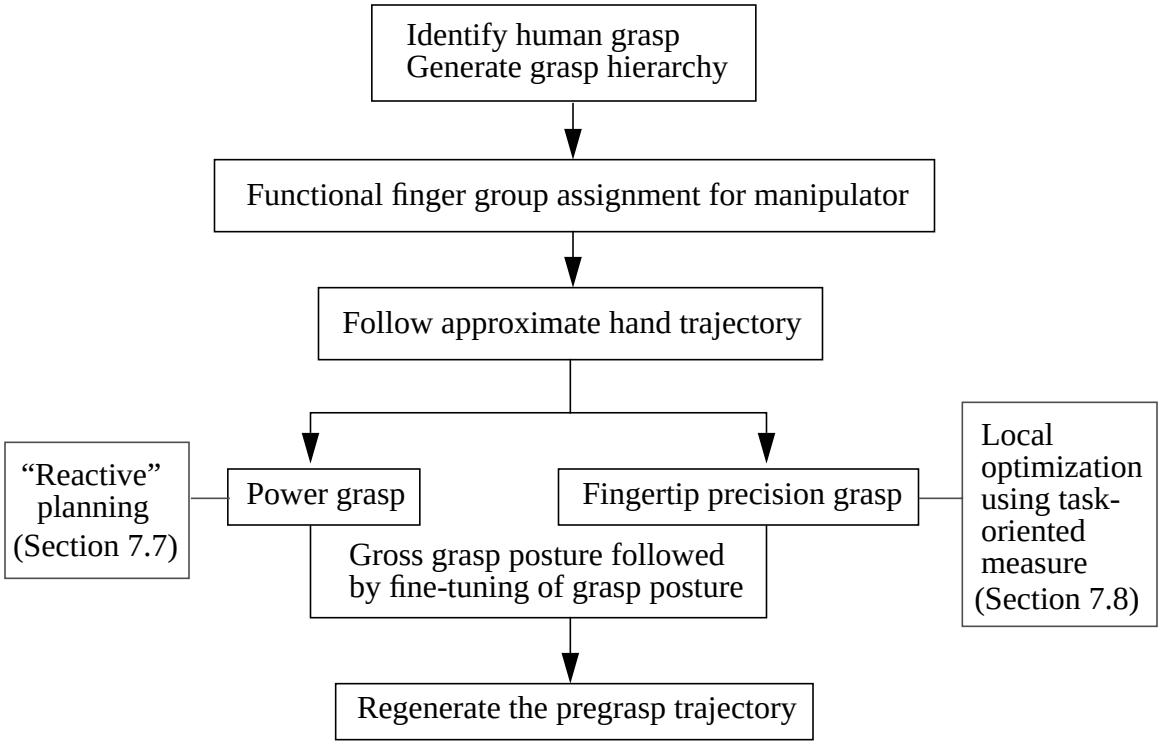


Fig. 74 **Scheme for generating the manipulator grasp and pregrasp trajectory**

Once local planning is done, starting with the pose and posture of the manipulator during the grasping stage, the pregrasp trajectory is then planned backwards, with the goals of a smooth and collision-free path. This path would closely follow that assumed by the human hand trajectory. Before we describe the local grasp planning approaches, we discuss a basic issue in all grasp and task planning – collision detection.

7.6 Collision Detection

In order to generate the grasps using the manipulator, collision detection between the fingers and fingers and object is required. To do this, the following was done:

- Object

  The object is represented by a collection of moderately dense surface points (with spacing of about 1-3 mm). Associated with each of these points is the object surface normal. These data are generated by a program which is in the same environment as the geometric modeler VANTAGE. The object normal is useful in determining the motions taken in planning the grasp.

- Fingers of manipulator

  Each finger is modeled by a conical frustrum with hemispherical caps at each end.

- Palm of manipulator

  The palm is modeled by a collection of convex polyhedra.

Checking for collision between the object and the fingers are done by checking for inclusion of each of the sampled points in the finger representation. This is similarly done for the palm. Checking for collision between finger segments are less straightforward: this is done by the following: To check if finger segment 1 collides with finger segment 2, approximate segment 1 as a union of spheres whose centers are along the segment spine. For each sphere, to check for intersection, shrink the sphere to a point, and expand segment 2 by the amount equal to the sphere radius. This reduces the problem of checking for point inclusion in the expanded segment representation. The entire procedure is subsequently repeated by switching segment 1 with segment 2 (i.e., approximating segment 2 as a union of spheres and checking these spheres for intersection with segment 1). This reduces the probability of error due to approximation.

Depending on the type of grasp, the approach in manipulator grasp planning differs. In the subsequent description of such mapping techniques, we concentrate on the most widely used grasps, namely the power grasp and the fingertip precision grasp.

7.7 Generating the Power Grasp

The primary purpose of the power grasp, in which the hand generally envelops the object, is to ensure a secure hold of the held object at the expense of dexterity. As such, we are primarily concerned with the manipulator providing complete or high kinematic restraint on the object. Hence, in the case of the power grasp, we can dispense with the third step of local grasp adjustment.

7.7.1 General Approach

The approach taken to generate the power or “enveloping” grasp follows that taken in “reactive” planning (similar to that described by Brock and Salisbury [18]). Note that this approach is taken in generating the grasp, and is not used in the actual execution of the grasp itself. The approach is characterized with the following phases:

1. *Aligning the dextrous hand with the approach vector assumed by the human hand just prior to grasping (approach alignment phase).*
2. *Translating the hand (preserving its orientation) towards the object until the palm touches the object (approach phase).*
3. *Adjust the hand to "fit" the object to the palm (adjustment phase).*
4. *Flex the fingers to envelope the object (wrapping phase).*

Central to the alignment motions of the dextrous hand is the *pivot point*. It is predefined to be a point on the surface of the palm between the fingers. Its frame is oriented such that its z-direction points away from the palm and the x-direction points from the thumb (or a primary finger/s) to the rest of the fingers.

7.7.2 Approach Alignment of Dextrous Hand

In order to minimize the problem of collision with the object during the pregrasp and post grasp phases, we first align the pivot frame of the dextrous hand to the approach vector assumed by the human hand just prior to grasping the object. This is known as the *approach alignment phase*. The approach vector is the translation component of the differential transformation ΔT, where $\Delta T = T_g \, T_{g\text{-}k}^{-1}$, where T_g is the transformation of the hand at the grasp frame and $T_{g\text{-}k}$ is the transformation of the hand *k* frames prior to the grasp frame (*k* is chosen to be 2).

Once the rotational transformation (about the pivot point) has been determined, the dextrous hand is rotated in steps, with the intermediate collision detection with the object. If collision is detected, the dextrous hand is translated or "repulsed" away (preserving its orientation) from the surface of object that makes contact with the hand. The normals of the surface or surfaces in question determine the direction of the repulsion vector.

7.7.3 Adjusting Hand and Enveloping Object

Once the desired orientation of the hand is assumed, planning proceeds with the following behaviors:

1. *Pure translation toward the object (approach phase)*

   The translational motion is dictated by attractive forces by the object on the *pivot point* on the dextrous hand. This behavior is active prior to any collision between the object and the manipulator palm.

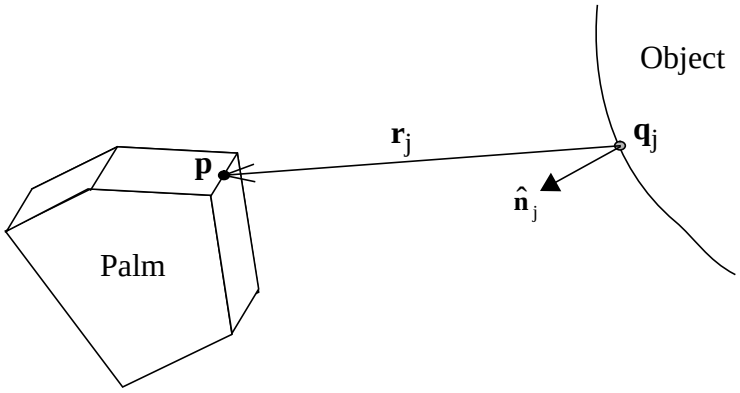


Fig. 75 **Determining direction of translation**

$\mathbf{p}$ is the pivot point on the manipulator palm, and $\mathbf{q}_j$ is the jth sampled object surface point. $\mathbf{r}_j$ is the vector from point $\mathbf{q}_j$ to $\mathbf{p}$, i.e., $\mathbf{r}_j = \mathbf{q}_j - \mathbf{p}$. $\hat{\mathbf{n}}_j$ is the object normal at $\mathbf{q}_j$. The motion at each stage is M:

$$M = \delta \frac{\mathbf{d}}{|\mathbf{d}|}$$

where δ is the step size in 3-D space,

$$\mathbf{d} = - \sum_{\hat{\mathbf{n}}_j \cdot \mathbf{r}_j > 0} \omega_j \mathbf{r}_j$$

ω_j is the weighting factor dependent on $|\mathbf{r}_j|$; the function used is $\omega_j = |\mathbf{r}_j|^{-2}$.

2. *Translation and rotation about the pivot point (adjustment phase)*

Once collision between the object and the manipulator palm has been detected, this behavior becomes active. The motion associated with this behavior is defined by both the point or points of contact and the entire object. The points of contact determine the motion which tend to push away the manipulator (repulsive forces) while the entire object tends to pull the manipulator (via the pivot point) towards the object. This has the property of trying to align the manipulator relative to the object prior to grasping.

It should be noted that while these behaviors are active, the posture of the manipulator is that of full extension of the fingers (for parallel jaw gripper, maximally distanced fingers).

Note: The circumflex (^) over the symbol indicates its normalized version.

The rotation about the pivot point is $\hat{Q}$, where

$$Q = \sum_{q_j \subset P} Q_j \quad ,$$

P is the palm, Q_j is the quaternion associated with the rotation due to the jth contact point and is given by

$$Q_j = \left[\cos\frac{1}{2}\theta_j, \mathbf{m}_j \sin\frac{1}{2}\theta_j \right]$$

$\theta_j = \delta/|\mathbf{r}_j|$ and $\mathbf{m}_j = -\mathbf{r}_j \times \mathbf{n}_j$.

3. *Flexion of fingers around the object (wrapping phase)*

   Once the manipulator has been adequately aligned with respect to the object, the manipulator would then proceed to wrap its fingers around it. This is done by flexing successively distal joints (starting with the most proximal joint) until contact is made between the associated link and the object.

The assumption taken in this technique is that the dextrous hand, when mapped to the coarse pregrasp trajectory, moves it to a reasonably near vicinity to the object at an appropriate posture. This should guarantee that, under the behaviors defined in the “reactive” planning (which is basically local), the manipulator posture should converge towards the desired power (enveloping) grasp.

This approach is a simple one, and is relatively independent of the manipulator. This approach is made possible by the cues given in the human execution of the same task. It should work, provided that the accuracy of the hand posture and position readings from the data acquisition system are reasonably accurate.

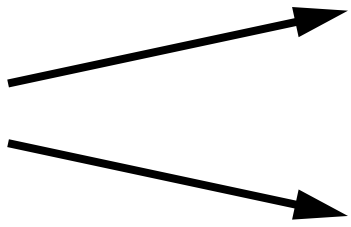


Fig. 76 **Results of mapping a cylindrical power grasp for Utah/MIT hand (top-right) and Salisbury hand (bottom-right)**

7.8 Generating Fingertip Precision Grasp

7.8.1 General Approach

The approach taken to plan the fingertip precision grasp is different than that for the power grasp, primarily because this precision grasp does not envelope the object. As with the power grasp, there are basically two stages:

1. *Initial grasp pose and posture determination phase*

   Let the (N+1) contact points on the object be denoted $\mathbf{p}_{c0}$, $\mathbf{p}_{c1}$, ... $\mathbf{p}_{cN}$. The initial grasp pose is determined by calculating the pose of the contact frame F_c and orienting the pivot frame F_{pivot} centered at the pivot point on the hand by a certain fixed transformation relative to F_c (Fig. 77).

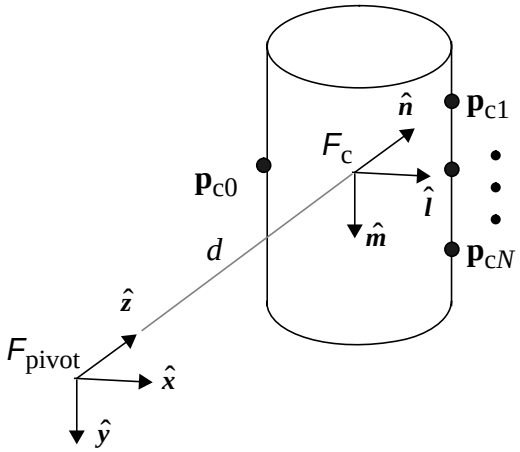


Fig. 77 **Orienting the pivot frame with respect to the contact frame**

Subsequently, the pose of the hand is searched at this neighhorhood until a kinematically feasible grasp is found. The contact frame F_c is centered about point **c**, defined to be

$$\mathbf{c} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(\mathbf{p}_{c0}+\mathbf{p}_{ci}) = \frac{1}{2}\left(\mathbf{p}_{c0}+\frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_{ci}\right)$$

The orientation of F_c is determined as follow (note that the caret ^ on a vector term indicates its normalized version):

$$\boldsymbol{l} = \sum_{i=1}^{N}(\mathbf{p}_{ci}-\mathbf{p}_{c0}) = \sum_{i=1}^{N}\mathbf{p}_{ci} - N\mathbf{p}_{c0}$$

If the best fit plane Π to the contact points is $\boldsymbol{r} \cdot \hat{\boldsymbol{v}} = a$, where $\boldsymbol{r}$ is any point on Π, and $\hat{\boldsymbol{v}}$ its unit normal, then

$$\hat{\boldsymbol{n}} = \begin{cases} \hat{\boldsymbol{v}}, & \text{if} \quad (\hat{\boldsymbol{v}} \times \hat{\boldsymbol{l}}) \cdot (\mathbf{p}_{cN} - \mathbf{p}_{c0}) > 0 \\ -\hat{\boldsymbol{v}}, & \text{otherwise} \end{cases}$$

and $\hat{\boldsymbol{m}} = \hat{\boldsymbol{n}} \times \hat{\boldsymbol{l}}$.

2. *Local grasp pose and posture adjustment phase*

This is done by locally optimizing a task-oriented criterion with respect to the pose and posture of the hand. The local adjustment is done at two levels; the two levels are associated with the optimization with respect to the pose (outer level) and with respect to the joint angles (inner level). Note that the inner level optimization require iterations only if redundancy of finger DOFs exists. For the Utah/MIT hand, each finger has a redundant DOF since there are 4 DOFs per finger with only the 3-D position of each fingertip being fixed. There is no redundancy of DOF for the three fingers of the Salisbury hand.

Let $\Delta_j = (\mathbf{t}_j, \mathbf{r}_j)$ be the vector representing the hand pose, Θ_{jk} be the vector of joint angles representing the hand posture, and $C()$ be the task-oriented criterion to be optimized. Then the optimization can be symbolically represented as:

$$\max_{\Delta_j} \left(\max_{\Theta_{jk}} C(\Delta_j, \Theta_{jk}) \right) = \max_{\Delta_j} C(\Delta_j, \Theta_{j,\,\mathrm{opt}})$$

The outer optimization level uses the Nelder-Mead simplex algorithm (see, for example, [102]), while the inner optimization is accomplished using Yoshikawa's scheme for optimizing manipulator postures with redundant DOFs [159].

For a given initial hand pose Δ_j, the locally optimal hand posture $\Theta_{j,\,\mathrm{opt}}$ is determined using the following pseudocode:

Repeat until either the joint angle limitations are exceeded or the magnitude of joint angle change vector $\dot{\Theta}\delta t$ is below a threshold {

$$\kappa_{jk} = \lambda \frac{\partial}{\partial \Theta} C(\Delta_j, \Theta) \Big|_{\Theta = \Theta_{jk}}$$

$$\begin{aligned} \dot{\Theta}_{jk} &= J^{+} \dot{\boldsymbol{x}} + (I - J^{+} J)\, \kappa_{jk} \\ &= (I - J^{+} J)\, \kappa_{jk} \end{aligned}$$

since $\dot{\boldsymbol{x}} = \mathbf{0}$, the contact points being fixed at each iteration.

$$\Theta_{jk} \leftarrow \Theta_{jk} + \dot{\Theta}_{jk} \delta t$$

}

where J is the hand Jacobian and J^+ is the pseudoinverse of the Jacobian, i.e.,

$$J^+ = J^T (JJ^T)^{-1}$$

7.8.2 Estimating Dextrous Hand-object Contact Points

In order to plan the robot fingertip grasp, we need to determine the points of contact between the robot hand and the object to be grasped. This is done by following these steps:

1. *Determine the rough positions of the human hand-object contact points.*

   Fit these contact points with a plane Π: $\boldsymbol{p} \cdot \hat{\boldsymbol{n}} = d$. This is done using the observation that human fingertip grasps result in contact points that approximately lie on a plane.

2. *Cull off planes and edges as potential grasping places by imposing the constraint that the associated object normal should not be too close in direction to* $\hat{\boldsymbol{n}}$.

   At each point on edge i (edge point) and point on face j (face point), let the associated normals be $\hat{\boldsymbol{n}}_{e_i}$ and $\hat{\boldsymbol{n}}_{f_j}$ respectively. Then, disregard points on edge i if $\left|\hat{\boldsymbol{n}}_{e_i} \cdot \hat{\boldsymbol{n}}\right| > \cos\theta_e$ and points on face j if $\left|\hat{\boldsymbol{n}}_{f_j} \cdot \hat{\boldsymbol{n}}\right| > \cos\theta_f$. Since we favor faces over edges, $\theta_e > \theta_f$, with $\theta_e = 75^o$ and $\theta_f = 45^o$.

3. *For each virtual finger with more than 1 real finger, fit a line to the human hand contact points, and space the hypothesized dextrous hand-object contact points equally along this line.*

4. *Find the object surface points nearest to the hypothesized dextrous hand-object points; for each face point, impose a margin* β_m *from the nearest edge (*β_m *= 1.0 cm) if possible; other wise arrange its position in the center between opposite edges.*

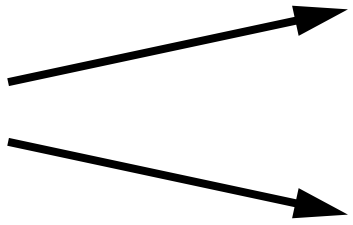


Fig. 78 **Results of mapping a fingertip precision grasp for Utah/MIT hand (top-right) and Salisbury hand (bottom-right)**

7.8.3 The Optimization Criterion

The precision grasp may be generated using a variety of task-oriented measures such as the manipulability measure [159], task compatability index [25], sum of force minimization [96], measures based on the task ellipsoid [85], among many others. We illustrate the notion of generating the precision grasp by local perturbation using Chiu's task compatability index [25].

Chiu [25] views the manipulator as a mechanical transformer with joint velocity and force as input and Cartesian velocity and force as output. The velocity and force transmission characteristics can be represented geometrically as ellipsoids defined as a function of the manipulator Jacobian. Let the kinematic transformation from joint space to task space be given by

$$\boldsymbol{x} = \boldsymbol{x}(\Theta)$$

where $\Theta = [\theta_1, \theta_2, ..., \theta_n]^T$ and $\mathbf{x} = [x_1, x_2, ..., x_m]^T$ are the joint and task coordinate vectors respectively. The Jacobian links the time differentials:

$$\dot{\boldsymbol{x}} = J(\Theta)\,\dot{\Theta}$$

where $J(\Theta)$ is the mxn manipulator Jacobian matrix, whose elements are $J_{i,j} = \partial x_i / \partial \theta_j$. The unit sphere in joint space $\Re^n$ defined by $\left\|\dot{\Theta}\right\| \leq 1$ is mapped into an ellipsoid in $\Re^m$ defined by

$$\dot{\boldsymbol{x}}^T \left(J(\Theta)\,J^T(\Theta)\right)^{-1} \dot{\boldsymbol{x}} \leq 1$$

This ellipsoid is variously called the manipulability ellipsoid [159] and the velocity ellipsoid [25]. Analogous to the velocity ellipsoid is the force ellipsoid obtained from the relation

$$\tau = J^T(\Theta)\,\boldsymbol{f}$$

where $\boldsymbol{f}$ is the force vector in task space and τ is the joint torque vector. The set of achievable force in $\Re^m$ subject to the constraint $\tau^T\tau = ||\tau||^2 \leq 1$ is the force ellipsoid defined by

$$\boldsymbol{f}^T \left(J(\Theta)\,J^T(\Theta)\right)\boldsymbol{f} \leq 1$$

The force transmission ratio α and velocity transmission ratio β along direction $\mathbf{u}$ in task space can be determined to be

$$\alpha = \left[\mathbf{u}^T (JJ^T)\,\mathbf{u}\right]^{-\frac{1}{2}}$$

and

$$\beta = \left[\mathbf{u}^T (JJ^T)^{-1}\mathbf{u}\right]^{-\frac{1}{2}}$$

respectively. Chiu defines the *compatibility index* as

$$C = \sum_i w_i \alpha_i^{\pm 2} + \sum_j w_j \beta_j^{\pm 2}$$

$$= \sum_i w_i \left[\mathbf{u}_i^{\mathrm{T}} \left(JJ^{\mathrm{T}} \right) \mathbf{u}_i \right]^{\mp 1} + \sum_j w_j \left[\mathbf{u}_j^{\mathrm{T}} \left(JJ^{\mathrm{T}} \right)^{-1} \mathbf{u}_j \right]^{\mp 1}$$

where the positive exponent of α_i or β_j is used for directions in which the magnitude is of interest, and the negative exponent is used for the directions in which accuracy is of interest. The *w*'s are weighting factors that indicate the relative magnitude or accuracy requirements in the respective task directions.

In our application, the finer force control directions are chosen to be along the object normals at the points of contact, while the finer velocity control directions are along the tangent plane defined by two spanning tangents. This is illustrated in Fig. 79.

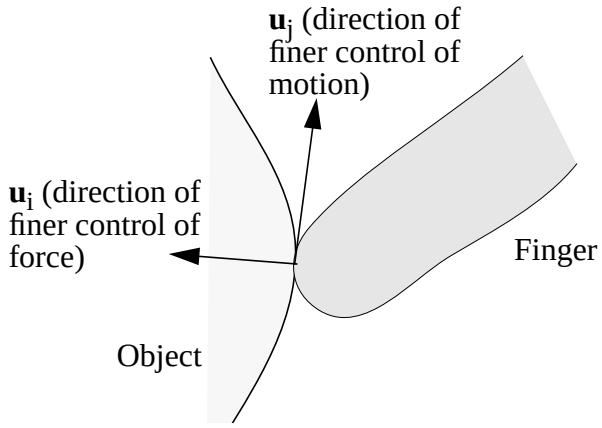


Fig. 79 **Contact between fingertip and object**

Hence we are trying to maximize

$$C = \sum_i w_i \mathbf{u}_i^{\mathrm{T}} \left(JJ^{\mathrm{T}} \right) \mathbf{u}_i + \sum_j w_j \mathbf{u}_j^{\mathrm{T}} \left(JJ^{\mathrm{T}} \right)^{-1} \mathbf{u}_j$$

with the w_i's set to 1.0 and w_j's to 10.0.

7.9 System Implementation

Robot System

The arm/hand system used to verify the grasp mapping technique is shown in Fig. 80. The arm used is the PUMA 560 arm while the dextrous hand used is the Utah/MIT hand. Both the arm and hand is controlled via Chimera, which is a real-time operating system for sensor-based control developed at Carnegie Mellon University [140]. The joint angles of the

arm and the hand are fed from the grasp simulator (Fig. 81) to the arm/hand system via Chimera.

Grasp Simulator

The grasp simulator was written in C and the interface in XView and PHIGS (Programmer's Hierarchical Interactive Graphics System). The robot arm and hand models were created in this environment; the object collision detection scheme described in section 7.6 is implemented in this grasp simulator as well, for the purpose of robot grasp planning.

The two types of grasp performed by the Utah/MIT hand are shown in Fig. 82.

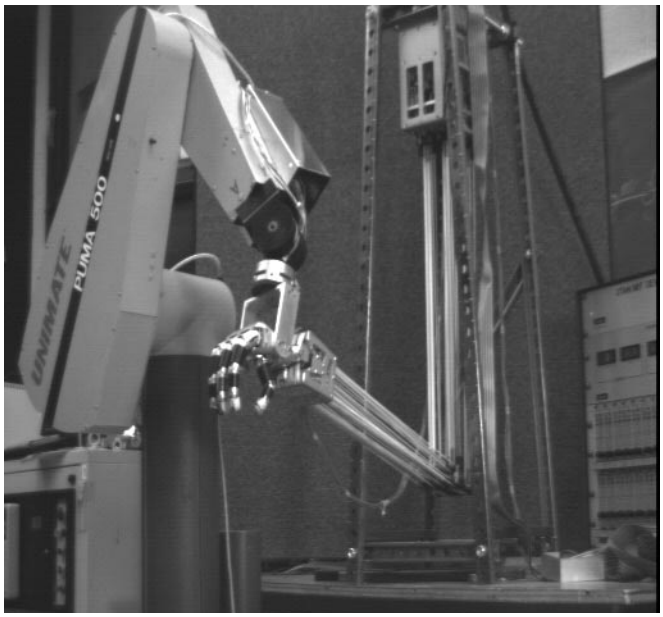


Fig. 80 **PUMA 560 and Utah/MIT hand system**

In general, depending on the task to be performed, a library of autonomous functions would be required. A good example is the task of turning a screw into a threaded hole; the exact motions used by a human in turning a screw may not be suitable for any specifically given manipulator. A skill library would be required – such as that described by Hasegawa, Suehiro, and Takase [47] – to execute fine manipulative actions not transferable directly by the human operator. This is still consistent with our notion of programming by human demonstration; the system basically recognizes certain actions and passes that high-level information to trigger the relevant skill library (which may include sensing strategies) to execute the task. The notion of skill library is not addressed in this thesis. Using a functional programming language is also consistent with the skill library; Speeter [139], for example, uses HPL (Hand Programming Language) for the Utah/MIT hand. In HPL, tasks are specified by a combination of "motor primitives," which are hand motion abstractions (e.g., a sequence of joint angle changes for the operations "close" or "open"). Michelman and Allen [108] pro-

gram the Utah/MIT hand to perform complex tasks by using sequential combinations of primitive manipulation functions. They consider only fingertip manipulations. The primitive manipulations include three translations (pinch, palmar, tranverse) and three rotation (log-roll, twiddle, pivot) strategies.

Fig. 81 **Simulator panel**

Fig. 82 **Cylindrical power grasp (left) and fingertip precision grasp (right) by the Utah/MIT hand**

7.10 Summary of Chapter

In this chapter, we have described our method of mapping the observed human grasp to that of the manipulator. The mapping is performed at two consecutive levels – the functional level and the physical level. In the functional level mapping, functionally equivalent groups of fingers are mapped. Once this is accomplished, at the physical level of mapping, the location and pose of the robot hand is refined to accommodate the different hand geometries in grasping the object.

While this method may not produce a globally optimal manipulator grasp (by some task-related measure), it greatly simplifies grasp planning. In addition, it illustrates our philosophy of easing the programming burden to mostly demonstrating the task; it reduces the complexity of programming and planning the task by taking advantage of the cues derived from observing a human.

Chapter 8

Robot Programming to Execution Example

In this chapter, we describe a complete cycle that starts from human demonstration of a task to the system, task analysis of observed data, and finally robot execution of the observed task. The human demonstration of the task is observed using Version 2 of the observation system, which allows more than one object manipulation at a time.

8.1 Image Depth Extraction

Depth of the scene at every frame is extracted using the algorithm described in Kang, *et al*. [74]. Noise (especially at the object borders) are removed by filtering out outliers. An example of extracted stereo data extracted from a scene is shown in Fig. 84(c).

8.2 Segmenting Task Sequence

The task sequence has been identified by the system as a 2-task, namely a task with two manipulation phases. The result of the temporal task segmentation is shown in Fig. 83. The task involved picking and placing a receptacle, followed by picking up a peg and inserting it into the hole of the receptacle.

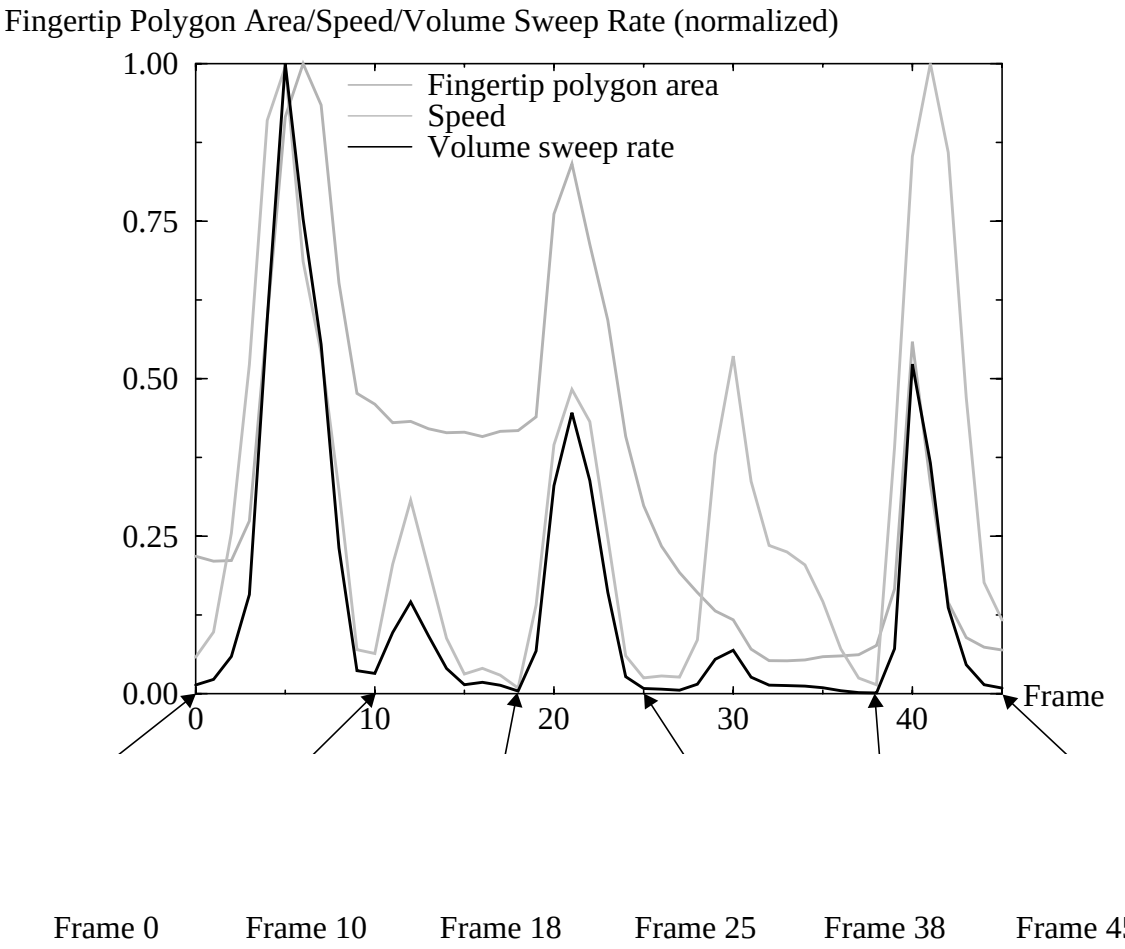
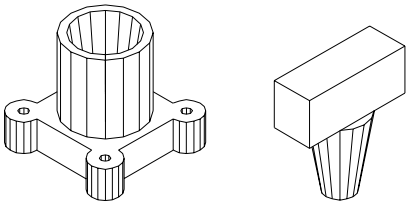


Fig. 83 **Recovered task breakpoints**

8.3 Identifying the Grasped Object

The initial gross positions of the objects were manually determined by the user; the 3DTM program subsequently refines these poses. The results of object localization are shown in Fig. 84.

(a)



(b) (c)

Fig. 84 **Object model fitting: (a) Geometric models of objects (receptacle, at left, and peg, at right), (b) superimposed object models in a scene, and (c) corresponding 3-D views**

8.4 Tracking 3-D Object

By using the *Cyber*Glove and Polhemus data and the results of temporal task segmentation, the object that is being moved during each manipulation phase can be automatically identified. The object is subsequently tracked using the approach described in section 6.5.2. The results of object tracking are shown in Fig. 85.

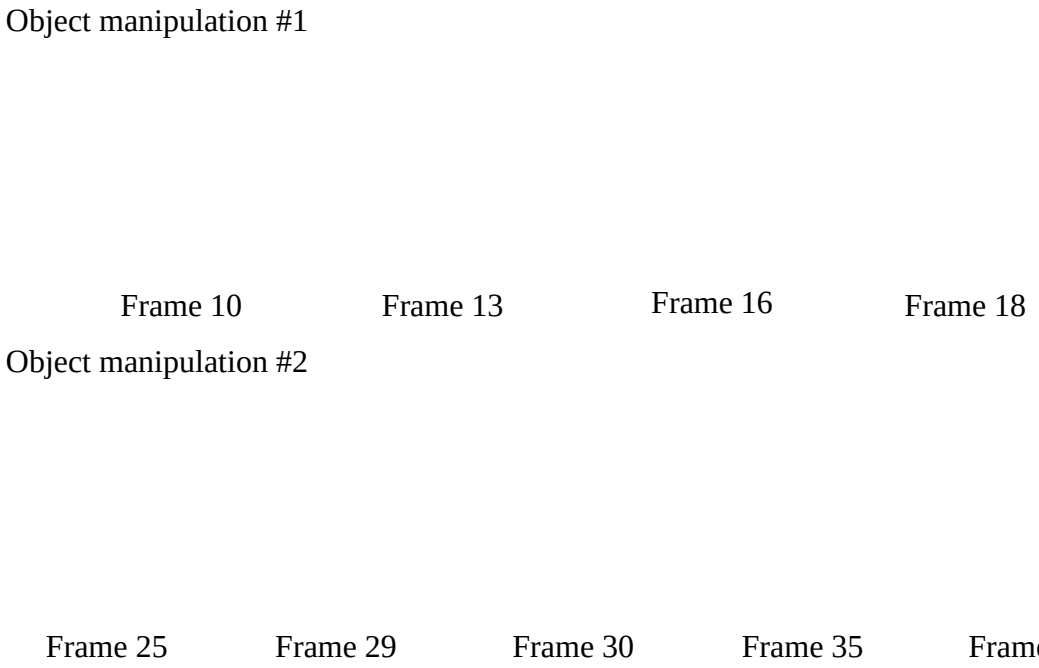


Fig. 85 **Object tracking results (selected frames shown). Note that significant object motion may occur between frames, e.g., between frames 29 and 30.**

8.5 Human Grasp Recognition

Once the object trajectory has been extracted, the two human grasps (the first with the receptacle and the second with the peg) have to be recognized. Prior to that, however, the hand poses have to be adjusted to cater for sensor inaccuracies. The results of hand adjustment are shown in Fig. 86 (first grasp) and Fig. 87 (second grasp).

(a) (b)

Fig. 86 **Hand pose (a) before, and (b) after adjustment (first grasp)**

(a) (b)

Fig. 87 **Hand pose (a) before, and (b) after adjustment (second grasp)**

The first grasp (with the receptacle) has been recognized as a “four-fingered disc grasp”:

```
-------------------------------------------------------
Best effective cohesive index = 0.487751
Virtual finger vf0 = { 0 }; Cohesive index = 1.000000
Virtual finger vf1 = { 1 }; Cohesive index = 1.000000
Virtual finger vf2 = { 2 3 }; Cohesive index = 0.696217
-------------------------------------------------------
Next best effective cohesive index = 0.469647
Virtual finger vf0 = { 0 }; Cohesive index = 1.000000
Virtual finger vf1 = { 3 }; Cohesive index = 1.000000
Virtual finger vf2 = { 1 2 }; Cohesive index = 0.621534
-------------------------------------------------------
Contact code is 011110 (no contact with palm and last finger)
N_active_fingers = 4; N_active_segments = 4

RMS error: prismatic_fit = 0.609335; disc_fit = 0.048098 (cm)
Grasp is FOUR-FINGERED DISC GRASP.

Number of virtual fingers = 3
Number of opposition spaces = 3
Virtual fingers involved in opposition space:
 opp0 : vf0, vf1; Opposition strength = 0.731697
 opp1 : vf0, vf2; Opposition strength = 0.974626
 opp2 : vf1, vf2; Opposition strength = 0.691342
```

Note that the extracted measurement of the last finger resulted in the analysis that it did not “touch” the object, when it actually did during the task. This is a problem due to both imperfect glove calibration and hand model.

The second grasp (with the peg) has been recognized as a “five-fingered prismatic grasp”:

```
---------------------------------------------------------
Best effective cohesive index = 0.610680
Virtual finger vf0 = { 0 }; Cohesive index = 1.000000
Virtual finger vf1 = { 1 2 3 4 }; Cohesive index = 0.745860
---------------------------------------------------------
Next best effective cohesive index = 0.531293
Virtual finger vf0 = { 0 }; Cohesive index = 1.000000
Virtual finger vf1 = { 4 }; Cohesive index = 1.000000
Virtual finger vf2 = { 1 2 3 }; Cohesive index = 0.899816
---------------------------------------------------------
Contact code is 111110 (no contact with palm)
N_active_fingers = 5; N_active_segments = 5

RMS error: prismatic_fit = 0.688206; disc_fit = 1.644156 (cm)
Grasp is FIVE-FINGERED PRISMATIC GRASP.

Number of virtual fingers = 2
Number of opposition spaces = 1
Virtual fingers involved in opposition space:
 opp0 : vf0, vf1; Opposition strength = 0.902369
```

8.6 Grasp Mapping

Subsequent to grasp recognition, the human grasps are mapped to manipulator grasps as shown in Fig. 88 and Fig. 89.

(a) (b)

Fig. 88 **Result of grasp mapping #1: (a) Human grasp, and (b) Utah/MIT grasp**

(a) (b)

Fig. 89 **Result of grasp mapping #2: (a) Human grasp, and (b) Utah/MIT grasp**

8.7 Robot Execution

Once the grasp mapping has been performed, the robot system comprising the PUMA arm and Utah/MIT hand is used to replicate the task as shown in Fig. 90.

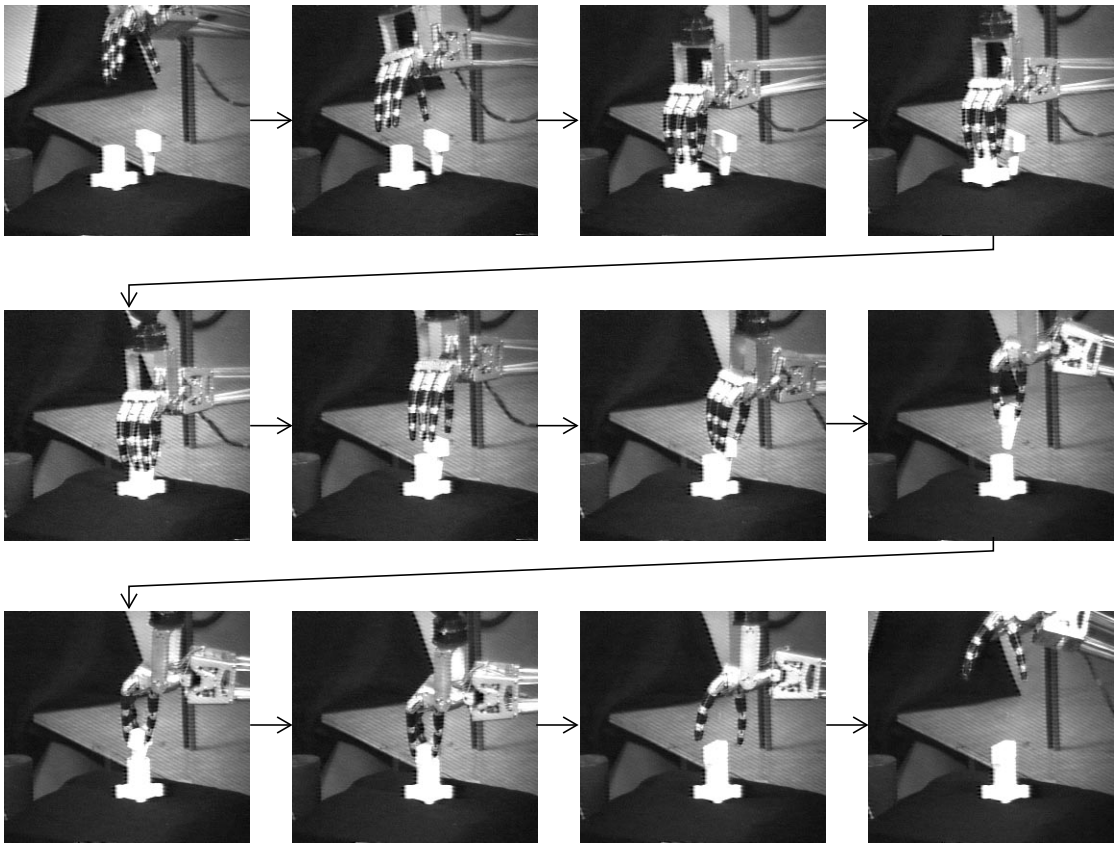


Fig. 90 **Different temporal snapshots of task execution by PUMA arm and Utah/MIT hand system**

Chapter 9

Conclusions

This thesis describes a programming-by-demonstration system that is capable of observing the performance of a task by a human operator, analyzing the stream of observation data, inferring the task descriptions, and planning the execution of the observed task. This system is fundamentally designed to relieve the programming load normally associated with hand-coding by relegating the programming effort to just demonstrating it in front of the system. An important feature of this system is the manipulator-independence of the task description. Planning is accomplished not merely at kinematic or joint level, thus allowing flexibility in using different general-purpose manipulators without the need to repeat task programming.

9.1 Programming-by-demonstration System – A Summary

The programming-by-demonstration system has four basic modules - the observation system, the task recognition module, the task translator, and the robot system.

9.1.1 Observation System

The observation or data acquisition system is responsible for acquiring perceptual streams of data associated with the recorded human execution of the task that is to be replicated. Two versions of this system have been described. The earlier version comprises a *Cyber*-Glove that measures human joint angles, a Polhemus device that tracks hand pose, and a light-stripe rangefinder that produces range data of the scene just prior to and subsequent to the execution of the task. A major problem with this setup is the inability of the system to obtain intermediate range data for more reliable object tracking. This is because the light-stripe rangefinder produces range data once every 10 secs or so.

Because of this limitation, a second setup version was used. This setup uses a multibaseline stereo system instead of the light-stripe rangefinder. The stereo system is capable of video rate (30 Hz) image capture; depth recovery, which yields dense range data, can then be

extracted subsequently. Because image acquisition has to be interlaced with *Cyber*Glove and Polhemus data capture, the sampling rate drops to about 4.5-5 Hz.

The quality and type of input data plays a crucial role in determining the complexity of precessing to yield the desired task description. The fundamental problem of trading off hardware solutions to software ones is evident. In the first version of the data acquisition system, dense range data are available only right before and right after the task execution. Object motion is inferred directly from measured human hand motion, and as a result, cannot be expected to be very accurate. A three-pass algorithm is used to accommodate both errors in hand and object poses at each frame of the task sequence. Since the rangefinder takes about 10 seconds to take a range image and reasonably accurate object localization is required after every object manipulation, each task is restricted to just one object manipulation.

In contrast to the first version, the second version of the data acquisition system provides a better wealth of information. This is because the stereo system that replaces the light-stripe rangefinder is capable of significantly faster rates of capture (about 5 Hz with data sampling from the *Cyber*Glove and Polhemus device). As a result, object motion can be tracked at each frame, dispensing with the 3-pass algorithm. In addition, each continuously executed task is not limited to just a simple object manipulation. However, all this is done at a much higher computational expense. This additional computation is primarily due to the range data extraction from stereo images and the recovery of object pose from the range data at every frame.

Is vision always needed? The work in this thesis requires knowledge of object pose at every frame in the sequence. Hand posture and pose is given by the *Cyber*Glove and Polhemus device respectively. All this is required for grasp recognition and for recovery of object motion trajectory. Vision is used because it appears to be the most apparent and direct method of extracting object pose, though it is by no means the only way.

9.1.2 Task Recognition Module

Once the preliminary stream of perceptual data associated with the execution of the task has been collected, it is then analyzed by the task recognition module to yield task descriptions. It first segments the stream of data into the general phases of pregrasp, grasp, and manipulation phases. The human grasp is then recognized; by analyzing the manipulation phase using the spectrogram, or spatially local Fourier analysis, repetitive actions such as screw turning can be detected and localized.

All this information, along with object motion information, is then passed to the task translator.

9.1.3 Task Translator

The task translator uses information supplied by the task recognition module to produce the robot task execution strategy. The task translator also uses the kinematic and geometric information associated with the robot system for this purpose.

It first plans the manipulator grasp by using human grasp descriptions extracted from observation. Grasp planning is done at two levels – the functional level, at which groupings of fingers (both human and robot) are linked, and the physical level, at which the manipulator grasp is planned using both object geometry information and a task-oriented local optimization scheme. In the proof-of-concept version, the planned manipulator path basically follows the trajectory of the human hand during the observed task execution.

9.1.4 Robot System

The system obviously need a physical device to execute the task once the observed task has been recorded and its associated stream of perceptual data analyzed. The robot system that is used as a testbed for the demonstrating the concepts developed in this thesis comprises the PUMA 560 arm and the Utah/MIT hand. Conceptually, this arm and hand may be substituted by another arm and/or general-purpose robot hand.

A major problem using the current robot system is its very limited effective workspace. This is mostly due to both the physical setup of the Utah/MIT hand and the hand being close to the maximum payload of the PUMA arm. As a result of this limited range of motion, several initial and final frames of the task are usually truncated. In addition, this imposes a restriction to the type of manipulative motion that can be replicated.

9.2 Thesis Contributions

This thesis has shown that a programming-by-demonstration system is implementable. It has detailed a set of techniques for data acquisition, data analyses to yield task descriptions, task planning based on these task descriptions, and actual robot execution.

The specific contributions of this thesis are:

1. *Development of a robust technique for segmenting a perceptual stream of data associated with the observed task into separate and meaningful subparts. A human hand motion feature called the* volume sweep rate *is introduced for this purpose. The segmentation algorithm preprocesses the data prior to further analyses designed to recover task descriptions.*
2. *Introduction of a grasp representation called the* contact web*, from which a grasp taxonomy and grasp recognition scheme are developed. The grasp recognition scheme uses a mapping algorithm to link physical human fingers to functionally grouped ones.*

3. *Development of a human grasp-to-manipulator grasp mapping scheme based on a two-tiered approach using the recovered human grasp descriptions. The two-tiered approach involves functional mapping, followed by physical mapping. The functional mapping makes the correspondences between the human and robot fingers, while the physical mapping adapts the robot hand to the object.*
4. *Development of a testbed for proof of concept that demonstrates the full cycle of programming by human demonstration, i.e., from data recording to robot execution.*

9.3 Future Directions

The work described in this thesis delineates an interesting direction in programming a robot. Programming a robot is reduced to merely demonstrating the task in front of the robot programming system. While significant progress has been made in the development of this system, there remain other issues to be explored to increase the level of sophistication in system capability.

9.3.1 Expansion of Grasp Taxonomy

The most direct way of increasing the repertoire of tasks that the system can infer from the recorded stream of perceptual data is to expand the grasp taxonomy. While many of the manufacturing grasps can be characterized under one of the current grasp taxonomy nodes, grasps that involves side finger contact with the grasped object (such as the hand posture assumed in writing) are not included. Furthermore, non-prehensile grasps as used in the act of pushing are not featured in the current taxonomy. While it may be desirable to include dynamic grasps in the extended taxonomy, the resulting taxonomy may not be of immediate use due to the limitations of the relatively low data sampling rate.

9.3.2 Interpretation of Force/tactile Information

At present, the system operates based solely on geometric and kinematic information. Having force information at the data collection stage would be helpful in guiding the planning of force strategy for robot task execution. It is not immediately obvious, however, that having force information would help significantly in grasp recognition, aside from verifying contact between finger segments and the grasped object.

9.3.3 Skill Library Development

Because of the kinematic dissimilarity between the human hand and the robot hand, the exact manner in which tasks are executed may not be replicated. This is especially so for tasks that involve fine manipulation. For instance, the manner in which human turns a screw into a threaded hole would generally be not optimal for a Utah/MIT hand to replicate exactly. As such, it is probably better if there are *a priori* designed “skill libraries” [47] associated with fine manipulation that are peculiar to the robot system. The appropriate skill

library can then be triggered and its parameter slots (such as diameter of screw head, depth of total vertical motion) can then be filled by the task translator based on the recovered task description.

9.3.4 Abilities to Learn and Adapt

In the current system, each single task is programmed independently of each other, even if there are similarities between the tasks. There is currently no cumulative learning ability, that is, the system is not able to adaptively finetune its performance as more demonstrations of the same task are performed. In addition, it may be desirable to able to have the system learn to associate the types of grasps with the objects in order to shorten the programming cycle.

Appendix A

Determining Transformation between Polhemus and Rangefinder Frames

This appendix describes how calibration is done to determine the transformation between the Polhemus device and light-stripe rangefinder frames. This calibration is done for Version 1 of the data acquisition system.

A.1 Polhemus Device Mounted at Back of Wrist

In order to merge the hand data from the *Cyber*Glove and Polhemus devices, and range data from the rangefinder, we first recovered the transformation between the *Cyber*Glove/Polhemus and rangefinder reference frames. This is done by laying the *Cyber*Glove/Polhemus devices on the table and taking their range image as well as the Polhemus readings. The required transformation between the two frames are determined from the recovered pose of the Polhemus device in the range image and the Polhemus readings.

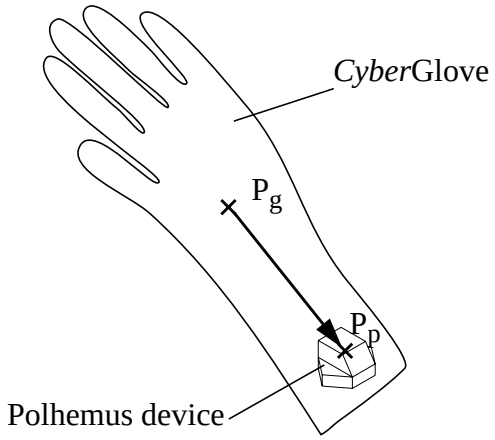


Fig. 91 **The *Cyber*Glove and Polhemus devices and their 3-D centroids**

The pose of the Polhemus device is extracted from the range image using the following two steps:

1. *Determine a rough estimate of the pose of the Polhemus device:*

   The 3-D centroid of the measurements of the *Cyber*Glove (P_g) is first calculated as shown in Fig. 91. Assuming that the Polhemus device is the brightest region in the image, the intensity image is thresholded and the 3-D centroid of the device (P_p) is found. P_p yields a coarse estimate of the device position; by calculating the vector difference between the two centroids (P_p-P_g) and normalizing it, we arrive at a coarse estimate of the device orientation (Fig. 92).

2. *Localization of the Polhemus device using the 3DTM algorithm [153]:*

   The 3DTM (3-D template matching) algorithm refines the pose estimate through minimization in a manner similar to deformable templates, active contours, and snakes [75][148]. In this case, the template is derived from the geometric model of the Polhemus device which is created using the geometric modeler Vantage [8]. The formulation of reducing the Euclidean distance between the image 3-D points and surface model points is based on the Lorentzian probability distribution; this distribution has the effect of lowering the sensitivity of localization error to object occlusion and extraneous range data. The final localized pose of the Polhemus model is shown superimposed on the image in Fig. 93.

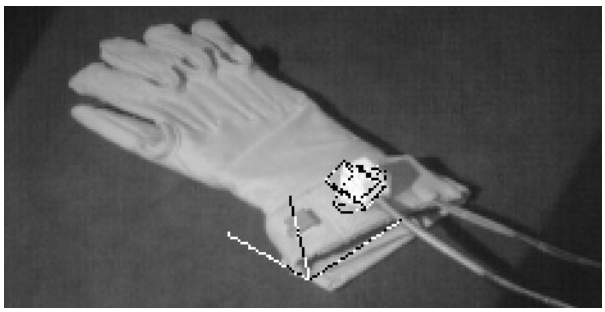


Fig. 92 **Location of the coarsely estimated pose of the Polhemus device**

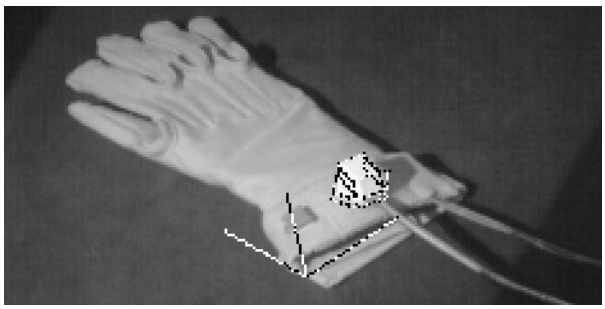


Fig. 93 **Final estimated pose of the Polhemus device**

Let the transformation corresponding to the recovered Polhemus pose in the rangefinder frame be denoted by ${}^{\text{cal}}_{\text{Range}}T$ and the transformation corresponding to the Polhemus readings be denoted by ${}^{\text{cal}}_{\text{Pol}}T$. Then the transformation that expresses the coordinates in the Polhemus frame in terms of those in the rangefinder frame is given by

$$ {}^{\text{Pol}}_{\text{Range}}T = {}^{\text{cal}}_{\text{Range}}T \cdot \left({}^{\text{cal}}_{\text{Pol}}T\right)^{-1} $$

A.2 Polhemus Device Mounted at Back of Hand

When the Polhemus device is moved to the back of the hand to reduce the amount of error propagation in the location of the fingers, estimating the initial pose of the Polhemus device is less straight-forward. The previous method of using the 3-D centroids of the *Cyber*Glove and Polhemus devices would not be reliable in this case (Fig. 91) due to their proximity to each other.

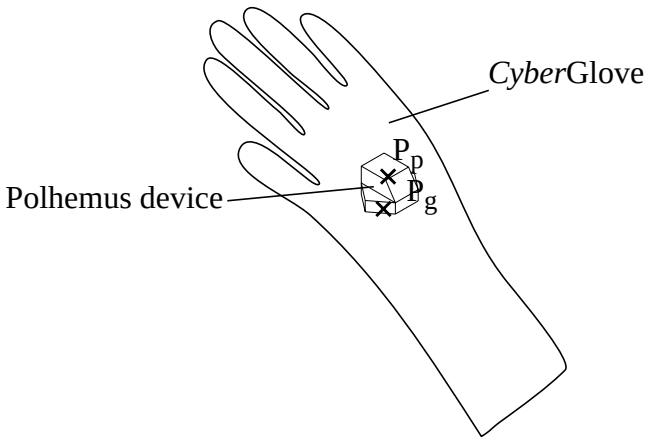


Fig. 94 **The *Cyber*Glove and Polhemus devices and their 3-D centroids**

The approach that we took comprises the following steps:

1. *Determine the approximate position of the Polhemus device*

   This is found by determining the 3-D centroid of the brightest region in the image (which is assumed to correspond to the Polhemus device).

2. *Determine from principal component analysis the 3-D major axis of the region occupied by both the* Cyber*Glove and Polhemus devices*

   This yields the estimated orientation of the Polhemus device. Note that this orientation could be anti-parallel to the correct initial orientation.

3. *Use the 3DTM algorithm to refine the pose of the Polhemus device in the range image*

   However, because of the ambiguity of the initial orientation, we use a two-pass approach: In the first pass, we input the original pose estimation into the program which outputs the refined pose and the average error. Subsequently we modify the refined pose by modifying the orientation by a rotation difference of 180^{o} and use this as input to the second pass. We use the refined pose which corresponds to the lower average error.

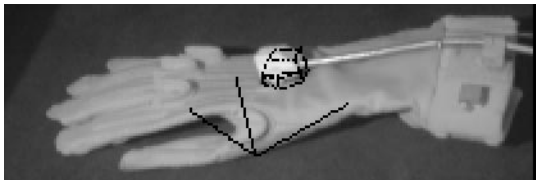


Fig. 95 **Initial pose of the Polhemus device**

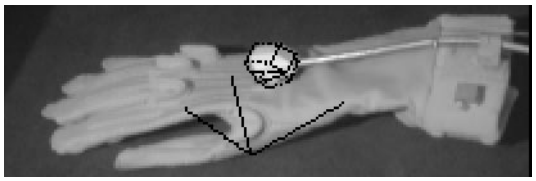


Fig. 96 **Pose immediately after the first pass (switch in orientation)**

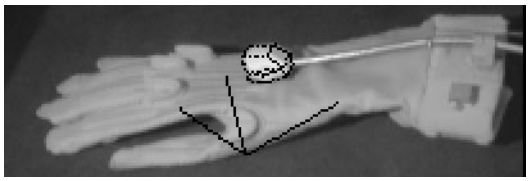


Fig. 97 **Final pose of Polhemus device**

Once the pose of the Polhemus device had been determined, the transformation between the range-finder and Polhemus frames are calculated as in the previous section.

A.3 Linear Interpolation of Polhemus-to-rangefinder Transform

The Polhemus device was originally mounted at the dorsal aspect of the wrist portion of the *Cyber*Glove. This has created serious inaccuracies in the hand and finger locations as the error accumulates from the wrist outwards. These errors include inaccurate hand dimensions and configurations, and errors in joint angle measurement. The Polhemus device uses an ac low-frequency, magnetic field technology to determine the position and orientation of a sensor in relation to a source; nearby ferromagnetic material causes field distortion which would then yield erroneous pose readings. Unfortunately, the equipment that we use has metallic chassis and support frames.

To reduce the inaccuracies due to the compounding effect of angular and configuration errors, we remounted the Polhemus device at the dorsal aspect of the hand. In addition, to compensate for the distorted magnetic field in the workspace (which causes the inaccuracies in the Polhemus readings), we calibrate the Polhemus device at several reasonably well-spaced places (eight) and interpolate between these calibration poses according to spatial proximity. The calibration poses are, however, restricted to those within both the camera and rangefinder views.

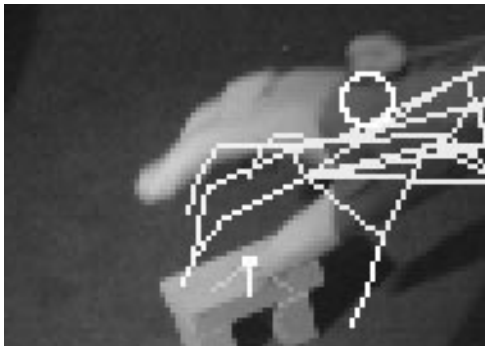
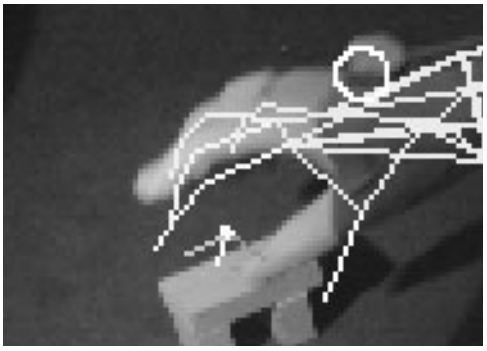
(a)  (b) 

Fig. 98 **Superimposed hand on image in a task sequence with (a) one calibration point, and (b) eight calibration points**

Fig. 98(a) and (b) show the effect of using several calibration poses as compared to just one.

Appendix B

Determining Transformation between Polhemus and Stereo Frames

The procedure described in this appendix is essentially the same as that for Version 1 of the data acquisition system in Appendix A. The differences are:

1. *The Polhemus device is too small (given the desired workspace size) to reliably extract stereo data from and compare with the geometric model. To obviate this problem, a larger calibration shell device is fitted over the Polhemus during calibration.*
2. *The active multibaseline stereo works well with uniformly reflective scene for recovery of dense data. As a result, unlike the procedure described in Appendix A, the calibration shell device cannot be assumed to be the brightest region in the scene.*

The principle for calibration is simple – to find the pose of the calibration shell device relative to the range data, and record the Polhemus reading at the same instant. The desired transformation between the Polhemus and stereo frames is a function of these two readings.

Because of point 2, in determining the pose of the calibration shell device, its initial gross pose is manually determined using two simultaneous interfaces as shown in Fig. 99. Fig. 99(a) shows four views of the object and scene points subsequent to manual object adjustment. Fig. 99(b) shows the superimposed model on one of the stereo camera views. The manual adjustment is facilitated by these simultaneous views. Fig. 100(a) and (b) show the result of applying the 3DTM program [153] on the manual object pose. As before, the Polhemus is attached to the back of the hand rather than to the back of the wrist.

(a) (b)

Fig. 99 **After manual gross pose localization: (a) Four 3-D views of object and scene, and (b) superimposed model on 2-D image.**

(a) (b)

Fig. 100 **After fine pose localization of object pose using 3DTM program: (a) Four 3-D views of object and scene, and (b) superimposed model on 2-D image.**

The transformation that links the Polhemus frame to the range frame can be expressed by the equation

$$
\begin{aligned}
{}^{\text{Pol}}_{\text{Range}}T &= {}^{\text{Shell-Vmodel`}}_{\text{Range}}T \cdot \left({}^{\text{Shell-Vmodel`}}_{\text{Pol-Vmodel`}}T\right)^{-1} \cdot {}^{\text{Pol`}}_{\text{Pol-Vmodel`}}T \cdot \left({}^{\text{Pol`}}_{\text{Pol}}T\right)^{-1} \\
&= {}^{\text{Shell-Vmodel`}}_{\text{Range}}T \cdot \left({}^{\text{Shell-Vmodel}}_{\text{Pol-Vmodel}}T\right)^{-1} \cdot {}^{\text{Pol}}_{\text{Pol-Vmodel}}T \cdot \left({}^{\text{Pol`}}_{\text{Pol}}T\right)^{-1}
\end{aligned}
$$

where the lower left subscript refers to the base (reference) frame and the upper left superscript refers to the object (Fig. 101). In the above equation and Fig. 101, Pol refers to the Polhemus device, Pol-Vmodel refers to the geometric model of the the Polhemus device, and Shell-Vmodel is the geometric model of the calibration shell.

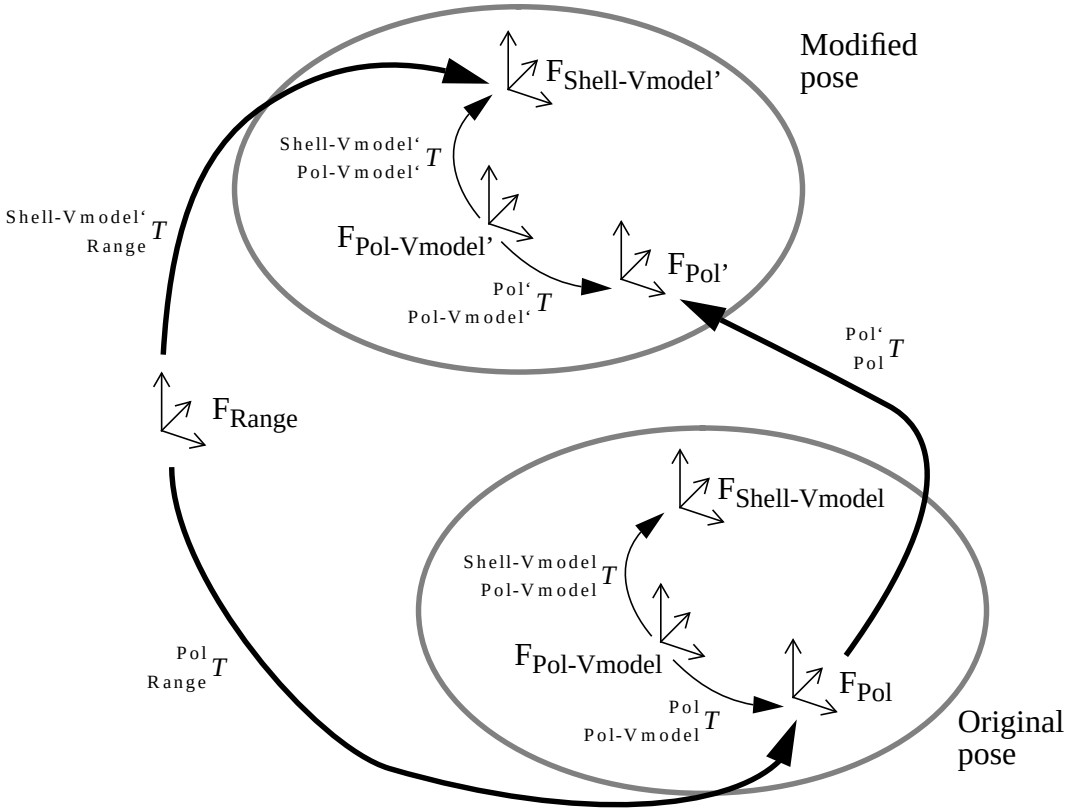


Fig. 101 **Transformations between range and model frames**

Appendix C

Calibrating Multi-camera System and Recovering Depth

C.1 Camera Alignment

In order to verge the cameras onto a volume of interest, we use a globe placed at the center of the volume. A globe is used since it is rotationally symmetric and thus should be look identical regardless of the camera view. Camera alignment involves adjusting the camera pose (location along the supporting bar and orientation) and camera setting (zoom, focus, and aperture). There are basically two sets of procedures in camera alignment. The first procedure is to first adjust the reference camera in order to get the reference image. The second procedure is to adjust the poses and settings of all the other cameras so that the globe as seen from all the cameras coincide. Note that absolute accuracy is not necessary since the camera calibration program (described in the following section) recovers the camera parameters reasonably accurately. The setup for camera alignment is shown in Fig. 102.

The steps for adjusting camera 1 (reference camera) pose and setting, and generating the reference image are:

1. *Attach video output of camera 1 to color monitor (line A or B)*
2. *Adjust pose of camera 1 until the globe is approximately in the center of the image (adjust tilt and pan of tripod stand if necessary).*
3. *Adjust camera settings (zoom, focus, etc.) of camera 1 until desired image quality and span of scene is achieved. This is the reference image.*

The steps for adjusting the other camera poses and settings are as follow:

1. *Connect video output of camera 1 to R (red) line of color monitor and that of camera 2 to G (green) line. Connect video output of camera 4 to sync line of color monitor.*
2. *Change the pose and settings of camera 2 so that the globes in the two superimposed images coincide.*
3. *Repeat for video outputs of camera 3 and camera 4 (for camera 4, use some other video output line as sync signal).*

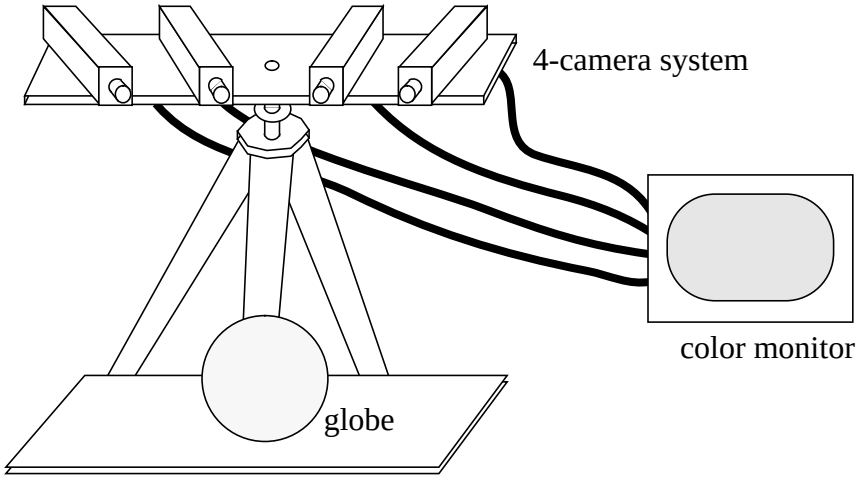
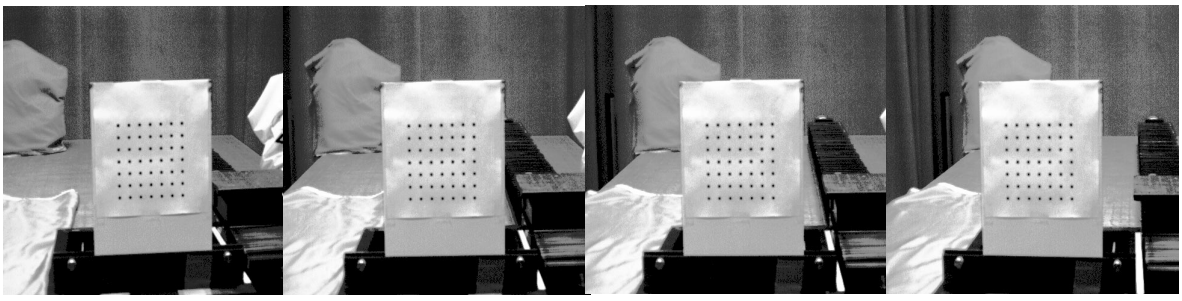


Fig. 102 **Setup for camera alignment (in a convergent configuration)**

C.2 Camera Calibration

The process of calibrating the camera configuration refers to the determination of the *extrinsic* (relative pose) and *intrinsic* (optic center offset, focal length and aspect ratio) camera parameters. The pinhole camera model is assumed in the calibration process. The origin of the verged camera configuration coincides with that of the leftmost camera.

A planar dot pattern arranged in a 7x7 equally spaced grid is used in calibrating the cameras; the images of this pattern is taken at various depth positions (five in our case). An example of a set of images taken by the camera system is shown in Fig. 103.



(a) (b) (c) (d)

Fig. 103 **Calibration images (equalized) taken from the convergent camera configuration ((a)-(d))**

The dots of the calibration pattern are detected using a star-shaped template with the weight distribution decreasing towards the center. The entire pattern is extracted and tracked from one camera to the next by imposing structural constraints of each dot relative to its neighbors, namely by determining the nearest and second nearest distances to another dot. This effectively filters out wrong dot candidates, as shown in Fig. 104.

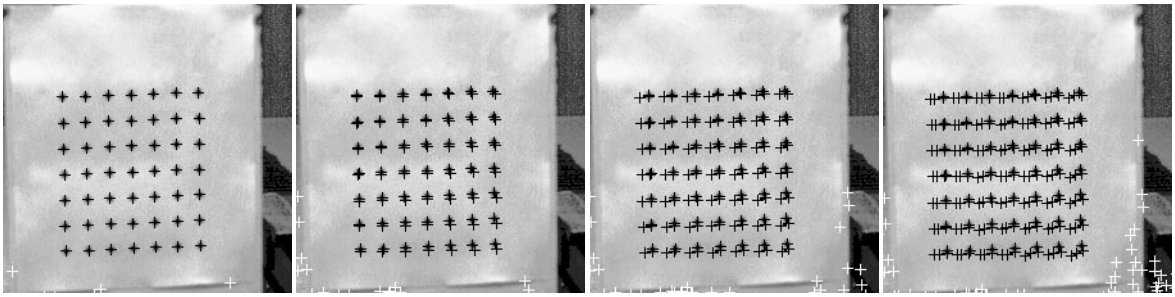


Fig. 104 **Detecting and tracking the calibration points (only part of the image associated with Camera 1 is shown). The black +'s are the detected points while the white +'s are the spurious and rejected points: (a) Points detected in image of Camera 1; (b) Points detected in images of Cameras 1 and 2; (c) Points detected in images of Cameras 1, 2, and 3; (c) Points detected in all the images.**

The simultaneous recovery of the camera parameters of all the four cameras can be done using the non-linear least-squares technique described by Szeliski and Kang [143]. The inputs and outputs to this module is shown in the simplified diagram in Fig. 105. An alternative would be to use the pairwise-stereo calibration approach proposed by Faugeras and Toscani [37].

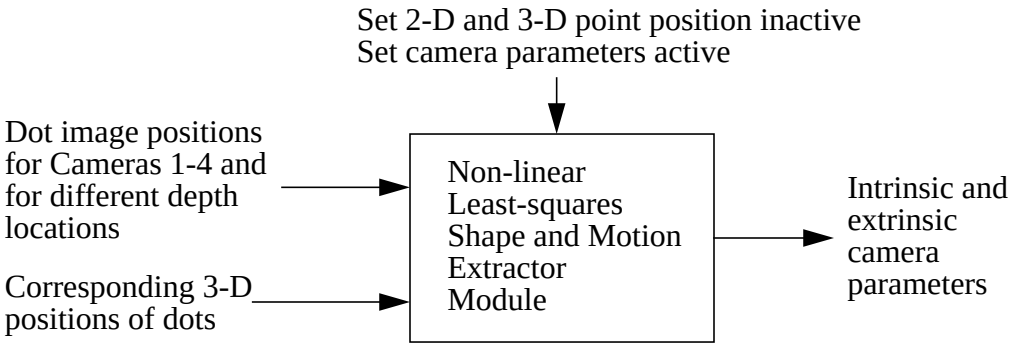


Fig. 105 **Non-linear least-squares approach to extraction of camera parameters**

C.3 Bandpass Filtering Images

The four cameras together with their video interface have slightly different output characteristics, resulting in different image intensities for the exact brightness in the scene. As a result, directly using the intensity images for depth extraction by local pixel comparison may not be optimal and result in a significant number of mismatches. (The depth extraction

scheme is briefly explained in the next subsection.) To mitigate the problem of mismatches, we first bandpass filter the images to reduce the low-frequency as well as very high-frequency components. The low frequency components are filtered out to reduce the dependency on the image intensity values; the very high frequency components are filtered out to reduce noise. An example of a bandpass filtered image used as input to the depth recovery program is shown in Fig. 106.

(a) (b)

Fig. 106 **Effect of bandfiltering image: (a) Original, and (b) Bandfiltered image**

C.4 Depth Recovery

If two cameras are not aligned parallel to each other, their associated epipolar lines are not parallel to the scan lines. This introduces extra computation to extract depth from stereo. To simplify and reduce the amount of computation, the preprocessing step of *rectification* can be carried out first. The process of rectification for a pair of images (given the camera parameters, either through direct or weak calibration) transforms the original pair of image planes to another pair such that the resulting epipolar lines are parallel and equal along the new scan lines. The rectification scheme used is described in Kang, *et al*. [74].

Once image rectification has been performed for the different pairs of images, we can then recover depth as shown in Fig. 107. P_j refers to the perspective projection matrix associated with the *j*th camera. M_k and N_k are the rectified perspective projection matrix associated with the *k*th rectified pair. H_{1p} is the homography (or linear projective correspondence) that maps pixels in the reference camera (Camera1) to those in the rectified image in the *p*th pair. K_{1q} is the homography that maps one of the rectified pair to the other (i.e., the mapping from pixel $\mathbf{v}_l$ to pixel $\mathbf{w}_l$ in Fig. 107). The mapping K_{1q} is a function of depth from Camera1.

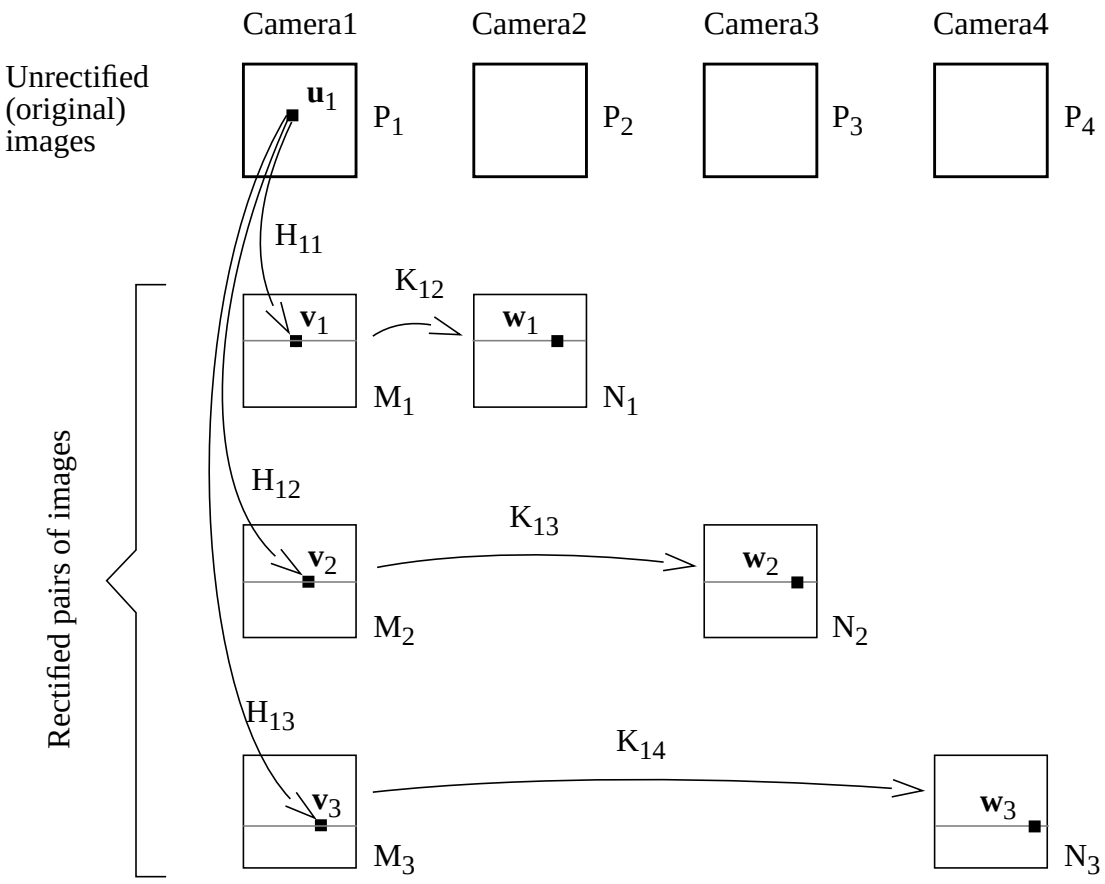


Fig. 107 **Depth recovery scheme**

The savings in computation subsequent to image rectification lies in pixels $\mathbf{v}_l$ and $\mathbf{w}_l$ lying on the same scanline (same y coordinates). The difference in the x coordinates between pixels $\mathbf{v}_l$ and $\mathbf{w}_l$ is the *disparity.* For each depth, the disparities for each pair is calculated and the error in local matching evaluated. The recovered depth at a given pixel is the depth that is associated with the least amount of error in local matching between pairs of rectified images.

We use an approximation to this scheme as shown in Fig. 108. Instead of starting from unrectified pixels and mapping it to respective pixels in the rectified images, we operate directly from the *rectified* image and propagate the mappings. The mappings L_{12} and L_{13} result in changes in both x and y coordinates; the mappings A_2 and A_3 attempts to preserve the y coordinates in the least squares sense (so that $\mathbf{w}_1$, $\mathbf{w}_2$, and $\mathbf{w}_3$ shown in Fig. 108 have y coordinates as close as possible). The mappings A_2 and A_3 are exact if and only if all the camera optical centers are colinear, which is not true in general. This approximation is good as long as the vergence angles between pairs of cameras are small. The vergence angle is typically less than 25^o between the cameras at the extremes.

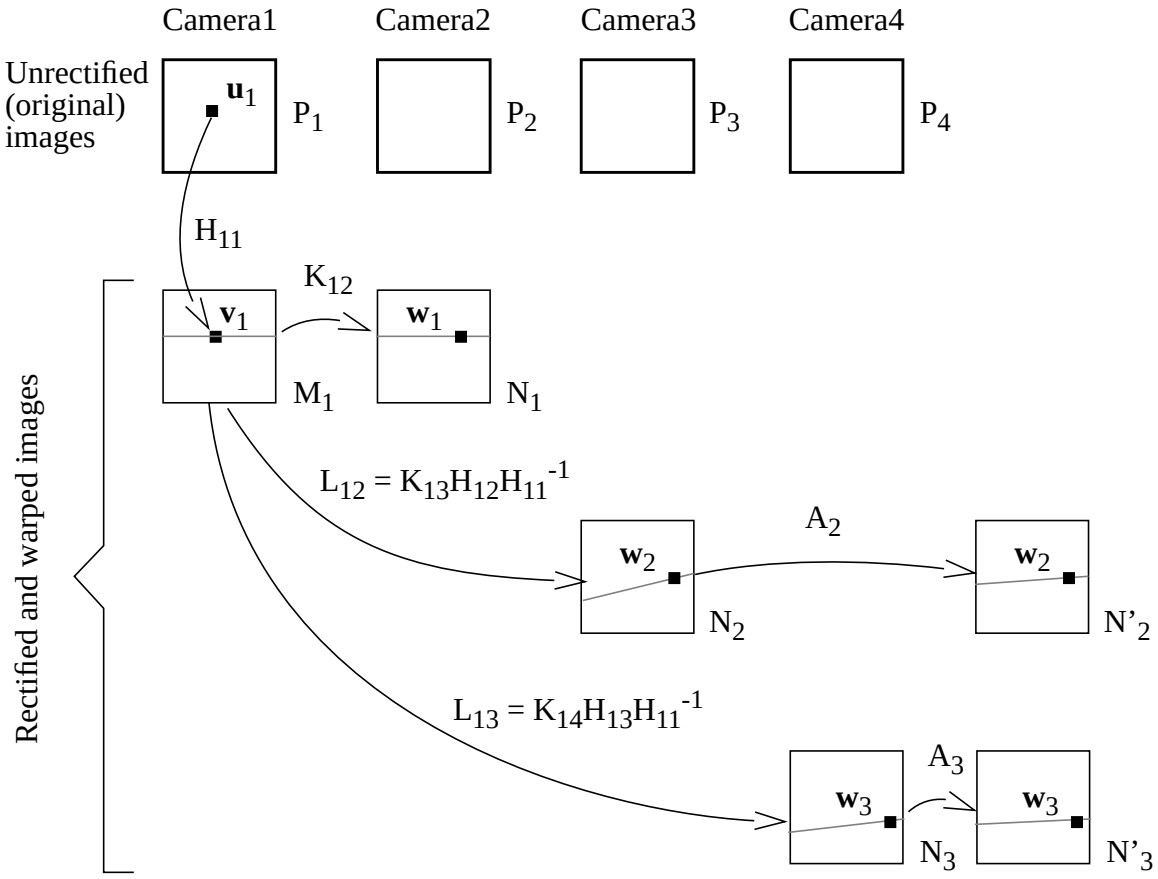


Fig. 108 **The approximate depth recovery scheme (compare this with Fig. 107)**

Details of the depth recovery scheme can be found in Kang, *et al*. [74].

Bibliography

[1] *3Space Isotrak User's Manual*, Polhemus, Inc., Jan. 1992.

[2] P. Allen, *Object Recognition Using Vision and Touch*, Ph.D. Thesis, University of Pennsylvania, 1985.

[3] K.N. An, E.Y. Chao, W.P. Cooney, and R.L. Linscheid, "Normative model of human hand for biomechanical analysis," *Journal of Biomechanics*, vol. 12, 1979, pp. 775-788.

[4] M.A. Arbib, T. Iberall, and D.M. Lyons, "Coordinated control programs for movements of the hand," *Hand Function and the Neocortex*, eds. A.W. Goodwin and T. Darian-Smith, Springer-Verlag, 1985, pp. 111-129.

[5] H. Asada and Y. Asari, "The direct teaching of tool manipulation skills via the impedance identification of human motions," *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1988, pp. 1269-1274.

[6] H. Asada and S. Hirai, "Towards a symbolic-level force feedback: Recognition of assembly process states," *Proc. 5th Int'l Symp. on Robotics Research*, 1989, pp. 341-346.

[7] H. Asada and J.-J.E. Slotine, *Robot Analysis and Control*, John Wiley and Sons, 1986.

[8] P. Balakumar, J.C. Robert, R. Hoffman, K. Ikeuchi, and T. Kanade, *VANTAGE: A Frame-based Geometric Modeling System - Programmer/User's Manual V2.0*, Tech. Rep. CMU-RI-TR-91-31, Carnegie Mellon University, Dec. 1991.

[9] C. Bard, J. Troccaz, and G. Vercelli, "Shape analysis and hand preshaping for grasping," *Proc. IEEE/RSJ Int'l Workshop on Intelligent Robots and Systems*, 1991, pp. 64-69.

[10] A.H. Barr, "Superquadrics and angle-preserving transformations," *IEEE Computer Graphics and Applications*, 1981, pp. 11-23.

[11] A. Bicchi, "Analysis and control of power grasping," *IEEE/RSJ Int'l Workshop on Intelligent Robots and Systems*, 1991, pp. 691-697.

[12] A. Bicchi, "Force distribution in multiple whole-limb manipulation," *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1993, vol. 2, pp. 196-201.

[13] A. Bicchi and C. Melchiorri, “Mobility and kinematic analysis of general cooperating robot systems,” *Proc. IEEE Int’l Conf. on Robotics and Automation*, 1992, pp. 421-426.

[14] L. Bologni, S. Caselli, and G. Vassura, “On grasp categorization for a dextrous robotic hand,” *Proc. Int’l Symp. on Sensorial Integration for Industrial Robots*, Zaragoza, Spain, Nov. 1989.

[15] S. Borkar, R. Cohn, *et al*., “Supporting systolic and memory communication in iWarp,” *Proc. 17th Int’l Symp. on Computer Architecture*, Seattle, WA, 1990, pp. 70-81.

[16] M. Brady, J.M. Hollerbach, T.L. Johnson, T. Lozano-Pérez, and M.T. Mason, *Robot Motion: Planning and Control*, MIT Press, 1982.

[17] J. Brinkman and H.G.J.M. Kuypers, “Cerebral control of contralateral and ipsilateral arm, hand and finger movements in the split-brain rhesus monkey,” *Brain*, vol. 96, 1973, pp. 663-674.

[18] D.L. Brock and J.K. Salisbury, “Implementation of behavioral control on a robot hand/arm system,” *Procs. 2nd Int’l Symposium on Experimental Robotics*, 1991, pp. 1-19.

[19] R.A. Brooks, “A robust layered control system for a mobile robot,” *IEEE Journal of Robotics and Automation*, vol. RA-2, 1986, pp. 14-23.

[20] B. Buchholz and T.J. Armstrong, “A kinematic model of the human hand to evaluate its prehensile capabilities,” *Journal of Biomechanics*, vol. 25, no. 2, 1992, pp. 149-162.

[21] H.J. Buchner, M.J. Hines, and H. Hemami, “A dynamic model for finger interphalangeal coordination,” *Journal of Biomechanics*, vol. 21, no. 6, 1988, pp. 459-468.

[22] T. Cao and A.C. Sanderson, “Task sequence planning in a robot workcell using AND/OR nets,” *Proc. IEEE Int’l Symp. on Intelligent Control*, 1991, pp. 13-15.

[23] S. Caselli, E. Faldella, B. Fringuelli, and F. Zanichelli, “A hybrid system for knowledge-based synthesis of robot grasps,” *Proc. IEEE/RSJ Int’l Workshop on Intelligent Robots and Systems*, 1993, pp. 1575-1581.

[24] S. Caselli, E. Faldella, and F. Zanichelli, “Grasp synthesis for a dextrous robotic hand using a multi-layer perceptron,” *6th Mediterranean Electrotechnical Conf.*, 1991, pp. 916-922.

[25] S.L. Chiu, “Task compatibility of manipulator postures,” *The Int’l Journal of Robotics Research*, vol. 7, no. 5, 1988, pp. 13-21.

[26] J.H. Connell, “A behavior-based arm controller,” *IEEE Trans. on Robotics and Automation*, vol. 5, no. 6, 1989, pp. 784-791.

[27] J.J. Craig, *Introduction to Robotics: Mechanics and Control*, Addison-Wesley, 1986.

[28] R.M. Crowder, "An anthropomorphic robotic end effector," *Robotics and Autonomous Systems*, vol. 7, no. 4, 1991, pp. 253-268.

[29] M.R. Cutkosky, "On grasp choice, grasp models and the design of hands for manufacturing tasks," *IEEE Trans. on Robotics and Automation*, vol. 5, no. 3, 1989, pp. 269-279.

[30] M.R. Cutkosky and R.D. Howe, "Human grasp choice and robotic grasp analysis," *Dextrous Robot Hands*, eds. S.T. Venkataraman and T. Iberall, Springer-Verlag, 1990, pp. 5-31.

[31] M.R. Cutkosky and I. Kao, "Computing and controlling the compliance of a robotic hand," *IEEE Trans. on Robotics and Automation*, vol. 5, no. 2, 1989, pp. 151-165.

[32] M.R. Cutkosky and P.K. Wright, "Modeling manufacturing grips and correlations with the design of robotic hands," *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1986, pp. 1533-1539.

[33] *CyberGlove^{TM} System Documentation*, Virtual Technologies, June 1992.

[34] N. Delson and H. West, "Robot programming by human demonstration: The use of human inconsistency in improving 3D robot trajectories," *Proc. IEEE/RSJ/GI Int'l Conf. on Intelligent Robots and Systems*, München, Germany, 1994, pp. 1248-1255.

[35] N. Delson and H. West, "Robot programming by human demonstration: The use of human variation in identifying obstacle free trajectories," *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1994, pp. 564-571.

[36] G.B. Dunn and J. Segen, "Automatic discovery of robotic grasp configuration," *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1988, pp. 396-401.

[37] O.D. Faugeras, and G. Toscani, "The calibration problem for stereo," *Proc. IEEE Int'l Conf. on Comp. Vision and Patt. Recog.*, 1986, pp. 15-20.

[38] R. Finkel, R. Taylor, R. Bolles, R. Paul, and J. Feldman, "AL: A programming system for automation," *Tech. Rep. AIM-177*, Stanford University, Artificial Intelligence Lab., 1974.

[39] M. Gardiner, "The superellipse: A curve that lies between the ellipse and the rectangle," *Scientific American*, vol. 213, 1965, pp. 222-234.

[40] H.E. Griffiths, *Treatment of the Injured Workman*, Lancet, 1943.

[41] D.D. Grossman and R.H. Taylor, "Interactive generation of object models with a manipulator," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-8, no. 9, 1978, pp. 667-679.

[42] E. Grosso and G. Vercelli, "Grasping strategies for reconstructed unknown 3D objects," *Proc. IEEE/RSJ Workshop on Intelligent Robots and Systems*, Osaka, Japan, 1991, pp. 70-75.

[43] W.A. Gruver, B.I. Soroka, J.J. Craig, and T.L. Turner, “Evaluation of commercially available robot programming languages,” *Proc. 13th Int’l Symp. on Industrial Robots*, 1983, pp. 12-58.

[44] T. Hamada, K. Kamejima, and I. Takeuchi, “Image based operation; A human-robot interaction architecture for intelligent manufacturing,” *Proc. 15th Conf. of IEEE Industrial Electronics Society*, 1989, pp. 556-561.

[45] T. Harima and H. West, “Natural robot programming system,” *Proc. IEEE/RSJ Int’l Conf. on Intelligent Robots and Systems*, 1992, pp. 750-755.

[46] H. Hashimoto and M. Buss, “Skill acquisition for the Intelligent Assisting System using Virtual Reality Simulator,” *Proc. 2nd Int’l Conf. on Artificial Reality and Tele-Existence (ICAT ‘92)*, Tokyo, Japan, 1992, pp. 37-46.

[47] T. Hasegawa, T. Suehiro, and K. Takase, “A model-based manipulation system with skill-based execution,” *IEEE Trans. on Robotics and Automation*, vol. 8, no. 5, 1992, pp. 535-544.

[48] S. Hirai and T. Sato, “Motion understanding for world model management of telerobot,” *Proc. 5th Int’l Symp. on Robotics Research*, 1989, pp. 5-12.

[49] J. Hong and X. Tan, “Calibrating a VPL DataGlove for teleoperating the Utah/MIT hand,” *Proc. IEEE Int’l Conf. on Robotics and Automation*, 1989, pp. 1752-1757.

[50] W. Iba, “Acquisition and improvement of human motor skills: Learning through observation and practice,” *Tech. Rep. RIA-91-29*, NASA Ames Research Center, Artificial Intelligence Research Branch, Oct. 1991.

[51] T. Iberall, “The nature of human prehension: three dextrous hands in one,” *Proc. IEEE Int’l Conf. of Robotics and Automation*, 1987, pp. 396-401.

[52] T. Iberall, “A ballpark approach to modelling human prehension,” *Proc. IEEE 1st Int’l Conf. on Neural Networks*, vol. 4, 1987, pp. 535-543.

[53] T. Iberall, “A neural network for planning hand shapes in human prehension,” *Proc. of American Control Conf.*, 1988, pp. 2288-2293.

[54] T. Iberall, G. Bingham, and M.A. Arbib, “Opposition space as a structuring concept for the analysis of skilled hand movements,” *Experimental Brain Research Series 15 – Generation and Modulation of Action Patterns*, eds. H. Heuer and C. Fromm, Springer-Verlag, 1986, pp. 158-173.

[55] T. Iberall, J. Jackson, L. Labbe, and R. Zampano, “Knowledge-based prehension: capturing human dexterity,” *Proc. IEEE Int’l Conf. of Robotics and Automation*, 1988, pp. 82-87.

[56] T. Iberall and C.L. MacKenzie, “Opposition space and human prehension,” *Dextrous Robot Hands*, eds. S.T. Venkataraman and T. Iberall, Springer-Verlag, 1990, pp. 32-54.

[57] K. Ikeuchi and S.B. Kang, "Assembly Plan from Observation," *AAAI Fall Symposium on Machine Learning in Computer Vision: What, Why and How?*, Raleigh, NC, 1993, pp. 115-119.

[58] K. Ikeuchi and T. Suehiro, "Towards an assembly plan from observation, Part I: Task recognition with polyhedral objects," *IEEE Trans. on Robotics and Automation*, vol. 10, no. 3, 1994, pp. 368-385.

[59] M. Inaba and H. Inoue, "Vision-based robot programming," *Proc. 5th Int'l Symp. on Robotics Research*, 1989, pp. 129-136.

[60] O.M. Ismaeil and R.E. Ellis, "Grasping using the whole finger," *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1994, pp. 3111-3116.

[61] S.C. Jacobson, J.E. Wood, D.F. Knutti, and K.B. Biggers, "The Utah/MIT dextrous hand: Work in progress," *Int'l Journal of Robotics Research*, vol. 3, no. 4, 1984, pp. 21-50.

[62] M. Jeannerod, "Intersegmental coordination during reaching at natural visual objects," *Attention and Performance IX*, eds. J. Long., and A. Baddley, Erlbaum, Hillsdale, NJ, 1981, pp. 153-168.

[63] M. Jeannerod, "The timing of natural prehension movements," *Journal of Motor Behavior*, vol. 16, no. 3, 1984, pp. 235-254.

[64] *Joint Motion: Method of Measuring and Recording*, American Academy of Orthopedic Surgeons, Chicago, 1965.

[65] M. Kaneko and K. Tanie, "Contact point detection for grasping of an unknown object using joint compliance," *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 1990, pp. 845-851.

[66] S.B. Kang and K. Ikeuchi, "Toward automatic robot instruction from perception – Temporal segmentation of tasks from human hand motion," conditionally accepted in *IEEE Trans. on Robotics and Automation*.

[67] S.B. Kang and K. Ikeuchi, "Robot task programming by human demonstration: Mapping human grasps to manipulator grasps," *Proc. IEEE/RSJ/GI Int'l Conf. on Intelligent Robots and Systems*, München, Germany, 1994, pp. 97-104.

[68] S.B. Kang and K. Ikeuchi, "Robot task programming by human demonstration," *Proc. Image Understanding Workshop*, Monterey, CA, Nov. 1994, pp. 1125-1134.

[69] S.B. Kang and K. Ikeuchi, "Determination of motion breakpoints in a task sequence from human hand motion," *Proc. IEEE Int'l Conf. on Robotics and Automation*, San Diego, CA, May 1994, pp. 551-556.

[70] S.B. Kang and K. Ikeuchi, "Grasp recognition and manipulative motion characterization from human hand sequences," *Proc. IEEE Int'l Conf. on Robotics and Automation*, San Diego, CA, May 1994, pp. 1759-1764.

[71] S.B. Kang and K. Ikeuchi, “Toward automatic robot instruction from perception - Recognizing a grasp from observation,” *IEEE Trans. on Robotics and Automation*, vol. 9, no. 4, 1993, pp. 432-443.

[72] S.B. Kang and K. Ikeuchi, “A grasp abstraction hierarchy for recognition of grasping tasks from observation,” *Proc. IEEE/RSJ Int’l Conf. on Intelligent Robots and Systems*, Yokohama, Japan, July 1993, pp. 621-628.

[73] S.B. Kang and K. Ikeuchi, “Grasp recognition using the contact web,” *Proc. IEEE/RSJ Int’l Conf. on Intelligent Robots and Systems*, Raleigh, NC, July 1992, pp. 194-201.

[74] S.B. Kang, J. Webb, C.L. Zitnick, and T. Kanade, “An active multibaseline stereo system with real-time image acquisition,” *Proc. Image Understanding Workshop*, Monterey, CA, Nov. 1994. A longer version is available as Tech. Rep. CMU-CS-94-167, Carnegie Mellon University, 1994.

[75] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *Int’l Journal of Computer Vision*, vol. 2, no. 1, 1987, pp. 322-331.

[76] *Knowledge Craft Manual - Vol. I: CRL Technical Manual*, Carnegie Group, Inc., 1989.

[77] J. Krumm and S.A. Shafer, “Local spatial frequency analysis of image texture,” *Proc. 3rd Int’l Conf. on Computer Vision*, 1990, pp. 354-358.

[78] T. Kuniyoshi, M. Inaba, and H. Inoue, “Teaching by showing: generating robot programs by visual observation of human performance,” *Proc. 20th Int’l Symp. on Industrial Robots*, 1989, pp. 119-126.

[79] M. Kuperstein, “Neural model of adaptive hand-eye coordination for single posture,” *Science*, vol. 239, 1988, pp. 1308-1311.

[80] P. Lammineur and O. Cornillie, *Industrial Robots*, Pergammon Press, 1984, pp. 43-54.

[81] J.M.F. Landsmeer, “Power grip and precision handling,” *Ann. Rheum. Dis.*, vol. 21, 1962, pp. 164-170.

[82] C. Laugier and J. Pertin-Troccaz, “ SHARP: A system for automatic programming of manipulation robots,” *Int’l Symp. on Robotics Research*, Gouvieux-Chantilly, France, 1985, pp. 125-132.

[83] D.G. Lawrence and H.G.J.H. Kuypers, “The functional organization of the motor system in the monkey. I. The effects of bilateral pyramidal lesions,” *Brain*, vol. 91, 1968, pp. 15-36.

[84] D.G. Lawrence and H.G.J.H. Kuypers, “The functional organization of the motor system in the monkey. II. The effects of the descending brainstem pathways,” *Brain*, vol. 91, 1968, pp. 15-36.

[85] Z. Li and S. Sastry, “Task oriented optimal grasping by multifingered robot hands,” *Proc. IEEE Int’l Conf. on Robotics and Automation*, 1987, pp. 389-394.

[86] D. Lian, S. Peterson, and M. Donath, “A three-fingered, articulated, robotic hand,” *Proc. 13th Int’l Symp. on Industrial Robots*, vol. 2, 1983, pp. 18/91-18/101.

[87] H. Liu, T. Iberall, and G.A. Bekey, “Neural network architecture for robot hand control,” *IEEE Control Sytems Magazine*, vol. 9, no. 3, 1989, pp. 38-43.

[88] H. Liu, T. Iberall, and G.A. Bekey, “The multi-dimensional quality of task requirements for dextrous robot hand control,” *Proc. IEEE Int’l Conf. on Robotics and Automation*, 1989, pp. 452-457.

[89] C. Long, P.W. Conrad, E.A. Hall, and S.L. Furler, “Intrinsic-extrinsic muscle control of the hand in power grip and precision handling,” *Journal of Bone and Joint Surgery*, Vol. 52-A, No. 5, 1970, pp. 853-867.

[90] T. Lozano-Pérez, “Robot programming,” *Procs. of the IEEE*, vol. 71, no. 7, 1983, pp. 821-841.

[91] T. Lozano-Pérez, “Automatic planning of manipulator transfer movements,” *IEEE Trans. on Systems, Man and Cybernetics*, vol. SMC-11, no. 10, 1981, pp. 681-689.

[92] T. Lozano-Pérez, J.L. Jones, E. Mazer, P.A. O’Donnell, *HANDEY - A Robot Task Planner*, MIT Press, 1992.

[93] D.M. Lyons, “A simple set of grasps for a dextrous hand,” *Proc. IEEE Int’l Conf. on Robotics and Automation*, 1985, pp. 588-593.

[94] C.L. MacKenzie, R.G. Marteniuk, C. Dugas, D. Liske, and B. Eickmeier, “Three-dimensional movement trajectories in Fitts’ task: Implications for control,” *Quarterly Journal of Experimental Psychology*, vol. 39A, 1987, pp. 629-647.

[95] C.L. MacKenzie and J. Van den Biggelaar, “The effects of visual information, object motion and size on reaching and grasping kinematics,” *Soc. for Neuroscience Abstracts*, vol. 13, part 1, 1987, pg. 351.

[96] X. Markenskoff and C.H. Papadimitriou, “Optimum grip of a polygon,” *The Int’l Journal of Robotics Research*, vol. 8, no. 2, 1989, pp. 17-29.

[97] X. Markenscoff, L. Ni, and C.H. Papadimitriou, “The geometry of grasping,” *The Int’l Journal of Robotics Research*, vol. 9, no. 1, 1990, pp. 61-74.

[98] R.G. Marteniuk and S. Athenes, “Characteristics of natural prehension movements for objects of varying size,” *Soc. for Neuroscience Abstracts*, vol. 12, part 2, 1986, pg. 970.

[99] R.G. Marteniuk, C.L. MacKenzie, M. Jeannerod, S. Athenes, and C. Dugas, “Constraints on human arm movement trajectories,” *Canadian Journal of Psychology*, vol. 41, no. 3, 1987, pp. 365-378.

[100] M.T. Mason, “Compliance and force control for computer controlled manipulators,” *IEEE Trans. Systems, Man and Cybernetics*, vol. 11, no. 6, 1981, pp. 418-432.

[101] M.T. Mason and J.K. Salisbury, *Robot Hands and the Mechanics of Manipulation*, MIT Press, 1985.

[102] J.H. Mathews, *Numerical Methods for Computer Science, Engineering, and Mathematics*, Prentice-Hall, 1987.

[103] T. Matsui and M. Tsukamoto, “An integrated robot teleoperation method using multi-media display,” *Proc. 5th Int’l Symp. on Robotics Research*, 1989, pp. 145-152.

[104] E.D. McBride, *Disability Evaluation*, T.B. Lippincott Co., 1942.

[105] B.J. McCarragher and H. Asada, “A discrete event approach to the control of robotic assembly tasks,” *Proc. IEEE Int’l Conf. on Robotics and Automation*, vol. 1, 1993, pp. 331-336.

[106] B. Mel, *Connectionist Robot Motion Planning: A Neurally Inspired Approach to Visually Guided Reaching*, Academic Press, 1991.

[107] K. Meyer and H.L. Applewhite, “A survey of position trackers,” *Presence*, vol. 1, no. 2, 1992, pp. 173-200.

[108] P. Michelman and P. Allen, “Forming complex dextrous manipulations from task primitives,” *Proc. IEEE Int’l Conf. on Robotics and Automation*, 1994, pp. 3383-3388.

[109] K. Mirza, M.D. Hanes, and D.E. Orin, “Dynamic simulation of enveloping power grasps,” *Proc. IEEE Int’l Conf. on Robotics and Automation*, 1993, vol. 2, pp. 430-435.

[110] P. Morasso, “Spatial control of arm movements,” *Experimental Brain Research*, vol. 42, no. 1, 1981, pp. 223-227.

[111] B.A. Myers, “Visual programming, programming by example, and program visualization: A taxonomy,” *Proc. CHI ‘86, Human Factors in Computing Systems*, 1986, pp. 59-66.

[112] J. Napier, “The prehensile movements of the human hand,” *Journal of Bone and Joint Surgery*, vol. 38B, no. 4, 1956, pp. 902-913.

[113] T.N. Nguyen and H.E. Stephanou, “A topological model of multifingered prehension,” *Proc. IEEE Int’l Conf. on Robotics and Automation*, 1989, pp. 446-451.

[114] T.N. Nguyen and H.E. Stephanou, “A computational model of prehensility and its application to dextrous manipulation,” *Proc. IEEE Int’l Conf. on Robotics and Automation*, 1991, pp. 878-883.

[115] V. Nguyen, “Constructing stable grasps in 3-D,” *Proc. IEEE Int’l Conf. on Robotics and Automation*, 1987, pp. 234-239.

[116] T. Okada, “Object-handling system for manual industry,” *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-9, no. 2, 1979, pp. 79-89.

[117] M. Okutomi and T. Kanade, “A multiple-baseline stereo,” *Proc. IEEE Int’l Conf. on Comp. Vision and Patt. Recog.*, 1991, pp. 63-69.

[118] L. Pao and T.H. Speeter, “Transformation of human hand positions for robotic hand control,” *Proc. IEEE Int’l Conf. on Robotics and Automation*, 1989, pp. 1758-1763.

[119] R.P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, MIT Press, 1981.

[120] K. Perlin, J.W. Demmel, and P.K. Wright, “Simulation software for the Utah/MIT dextrous hand,” *Robotics and Computer-Integrated Manufacturing*, vol. 5, no. 4, 1989, pp. 281-292.

[121] J. Pertin-Troccaz, “Grasping: A state of the art,” *The Robotics Review 1*, eds. O. Khatib, J.J. Craig, and T. Lozano-Pérez, MIT Press, Cambridge, MA, 1989, pp. 71-98.

[122] N. Pollard, “Planning grasps for a robot hand in the presence of obstacles,” *Proc. Int’l Conf. on Robotics and Automation*, 1993, pp. 723-728.

[123] P.K. Pook and D.H. Ballard, “Deictic teleassistance,” *Proc. IEEE/RSJ/GI Int’l Conf. on Intelligent Robots and Systems*, München, Germany, 1994, pp. 245-252.

[124] P.K. Pook and D.H. Ballard, “Recognizing teleoperated manipulations,” *Proc. IEEE Int’l Conf. on Robotics and Automation*, vol. 2, 1993, pp. 578-585.

[125] K. Rao, G. Medioni, H. Liu, and G.A. Bekey, “Shape description and grasping for robot hand-eye coordination,” *IEEE Control Systems Magazine*, vol. 9, no. 2, 1989, pp. 22-29.

[126] K. Rao, G. Medioni, H. Liu, and G.A. Bekey, “Robot hand-eye coordination: Shape description and grasping,” *Proc. Int’l Conf. on Robotics and Automation*, 1988, pp. 407-411.

[127] J. Rasmussen, “Skills, rules, and knowledge: Signals, signs, and symbols, and other distinctions in human performance models,” *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-13, no. 3, 1983, pp. 257-266.

[128] D. Reynaerts and H. Van Brussel, “Two-fingered full envelope dexterous manipulation,” *Proc. IEEE Int’l Conf. on Robotics and Automation*, 1993, vol. 2, pp. 436-441.

[129] H. Rijpkema and M. Girard, “Computer animation of knowledge-based human grasping,” *Computer Graphics*, Vol. 25, No. 4, 1991, pp. 339-348.

[130] J.M. Rubin and W.A. Richards, *Boundaries of visual motion*, Tech. Rep. AIM-835, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Apr. 1985.

[131] M. Salganicoff, *Learning and Forgetting for Perception-Action: A Projection Pursuit and Density Adaptive Approach*, Ph.D. Thesis, University of Pennsylvania, 1992.

[132] J. Salisbury, "Whole arm manipulation," *Proc. 4th Int'l Symp. on Robotics Research*, 1987, pp. 183-189.

[133] G. Schlesinger, "Der mechanische aufbau der künstlichen glieder," *Ersatzglieder und Arbeitshilfen fur Kriegsbeschadigte und Unfallverletzte*, eds. Borchardt, *et al*., Springer, 1919, pp. 321-699.

[134] D.A. Seres, R. Kelley, and J.R. Birk, "Visual robot instruction," *Proc. 5th Int'l Symp. on Industrial Robots*, 1975, pp. 113-126.

[135] B. Shepherd, "Applying visual programming to robotics," *Proc. IEEE Int'l Conf. on Robotics and Automation*, vol. 2, 1993, pp. 707-712.

[136] T.M. Sobh and R. Bajcsy, "Visual observation under uncertainty as a discrete event process," *Proc. 11th Int'l Conf. on Pattern Recognition*, vol. 1, 1992, pp. 429-432.

[137] F. Solina and R. Bajcsy, "Recovery of parametric models from range images: The case for superquadrics with global deformations," *IEEE Trans. on Patt. Analy. and Mach. Intell.*, vol. 12, no. 2, 1990, pp. 131-147.

[138] C. Sollerman, "Grip function of the hand: Analysis, evaluation and a new method," *Hand Surgery*, Dept. of Orthopaedic Surgery, Sahlgren Hospital, University of Goteberg, Goteberg, Sweden, 1980.

[139] T.H. Speeter, "Transforming human hand motion for telemanipulation," *Presence*, vol. 1, no. 1, 1992, pp. 63-79.

[140] D.B. Stewart and P.K. Khosla, *Chimera 3.1, The Real-Time Programming Operating System for Reconfigurable Sensor-Based Control Systems*, Carnegie Mellon University, 1993.

[141] T. Suehiro and K. Ikeuchi, "Towards an assembly plan from observation, Part II: Correction of motion paramters based on face contact constraints," *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 1992, pp. 2095-2102; a longer version is available as *CMU Tech. Rep. CMU-CS-91-168*, 1991.

[142] P.D. Summers and D.D. Grossman, "XPROBE: An experimental system for programming robots by example," *The Int'l Journal of Robotics Research*, vol. 3, no. 1, 1984, pp. 25-39.

[143] R. Szeliski and S.B. Kang, "Recovering 3D shape and motion from image streams using non-linear least squares," *Journal of Visual Communication and Image Representation*, vol. 5, no. 1, 1994, pp. 10-28.

[144] T. Takahashi and H. Ogata, "Robotic assembly operation based on task-level teaching in virtual reality," *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1992, pp. 1083-1088.

[145] T. Takahashi, H. Ogata, and S. Muto, "A method for analyzing human assembly operations for use in automatically generating robot commands," *Proc. IEEE Int'l Conf. on Robotics and Automation*, vol. 2, 1993, pp. 695-700.

[146] M. Tan, "CSL: A cost-sensitive learning system for sensing and grasping objects," *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1990, pp. 858-863.

[147] C.L. Taylor and R.J. Schwarz, "The anatomy and mechanics of the human hand," *Artificial Limbs*, No. 2, 1955, pp. 22-35.

[148] D. Terzopoulos, A. Witkin, and M. Kass, "Constraints on deformable models: Recovering 3D shape and nonrigid motion," *Artificial Intelligence*, vol. 36, 1988, pp. 91-123.

[149] R. Tomovic, G.A. Bekey, and W.J. Karplus, "A strategy for grasp synthesis with multifingered robot hands," *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1987, pp. 83-89.

[150] J.C. Trinkle, R.C. Ram, A.O. Farahat, and P.F. Stiller, "Dexterous manipulation planning and execution of an enveloped slippery workpiece," *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1993, vol. 2, pp. 442-448.

[151] Y. Uno, N. Fukumura, R. Suzuki, and M. Kawato, "Integration of visual and somatosensory information for preshaping hand in grasping movements," *Advances in Neural Information Processing Systems*, vol. 5, 1993, pp. 311-318.

[152] J.A. Webb, T. Warfel, and S.B. Kang, *A Scalable Video Rate Camera Interface*, Tech. Rep. CMU-CS-94-192, Carnegie Mellon University, 1994.

[153] M.D. Wheeler and K. Ikeuchi, "Sensor modeling, probabilistic hypothesis generation, and robust localization for object recognition," *Proc. 2nd CAD-Based Vision Workshop*, 1994, pp. 46-53.

[154] P.M. Will and D.D. Grossman, "An experimental system for computer controlled mechanical assembly," *IEEE Trans. Comput.*, vol. 24, no. 9, 1975, pp. 879-888.

[155] A.M. Wing and C. Fraser, "The contribution of the thumb to reaching movements," *Quarterly Journal of Experimental Psychology*, vol. 35A, 1983, pp. 297-309.

[156] A.M. Wing, A. Turton, and C. Fraser, "Grasp size and accuracy of approach in reaching," *Journal of Motor Behavior*, vol. 18, no. 3, 1986, pp. 245-260.

[157] J.D. Wolter, R.A. Volz, and A.C. Woo, "Automatic generation of gripping positions for assembly," *Robot Grippers*, eds. D.T. Pham and W.B. Heginbotham, Springer-Verlag, 1986, pp. 55-74.

[158] W.I. Yared and T.B. Sheridan, “Plan recognition and generalization in command languages with application to telerobotics,” *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 21, no. 2, 1991, pp. 327-338.

[159] T. Yoshikawa, “Manipulability of robotic mechanisms,” *The Int’l Journal of Robotics Research*, vol. 4, no. 2, 1985, pp. 3-9.